WebQuizz: Herramienta del profesor para la App Quizz

TRABAJO DE FIN DE GRADO DEL
GRADO EN INGENIERÍA DE SOFTWARE

Álvaro Navas Sánchez
Iván Quirós Fernández-Montes

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Junio 2018

Documento maquetado con TEXIS v.1.0+.

WebQuizz: Herramienta del profesor para la App Quizz

Trabajo de Fin de Grado en Ingeniería de Software

Álvaro Navas Sánchez
Iván Quirós Fernández-Montes

Dirigido por el doctor

Gonzalo Méndez Pozo

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Junio 2018

Resumen

El hardware es lo que hace a una máquina rápida; el software es lo que hace que una máquina rápida se vuelva lenta

Craig Bruce en 1990

Hoy en día la tecnología es algo que utilizamos casi sin darnos cuenta en nuestra vida cotidiana. Como tal, es muy importante hacer uso de los recursos que nos provee. De cara a poder utilizar estas herramientas, no basta con poner las mismas a disposición de todos; sino que es necesario el adaptarlas y enseñar a usarlas a los destinatarios de las mismas.

El objetivo fundamental de este Trabajo de Final de Grado es complementar la labor de los compañeros que hicieron la aplicación de Android *Quizz* el año pasado, dotando al profesor y a los propios usuarios de la misma de herramientas que les permitan gestionar el contenido de la propia aplicación. Para ello, este trabajo se centrará en 2 aspectos fundamentales:

- El desarrollo de un portal web para alumnos y profesores. Éste permitirá a los alumnos enviar contenido para añadir a la aplicación *Quizz* que, tras la consiguiente aprobación de su profesor, pueda pasar a formar parte de los juegos de la misma. Por otra parte, los profesores podrán gestionar las peticiones de adición de contenido por parte de sus alumnos y ver estadísticas del uso de la aplicación por parte de ellos de cara a orientarles o ayudarles a mejorar en lo que más problemas tengan.

- El desarrollo de un parseador que permita a los alumnos, a través de un documento Word con un determinado formato muy específico, subir ejercicios de los distintos juegos. Estos documentos Word deberán enviárselos al profesor que, tras hacer las correcciones que crea oportunas, los subirá al portal web, donde añadirá los ejercicios a la aplicación.

Palabras clave

El ordenador nació para resolver problemas que antes no existían

Bill Gates, cofundador de Microsoft

- Aplicación web
- Parseador
- Android
- Aprendizaje
- Juego
- Sistema de gestión de contenidos

Abstract

No temo a los ordenadores; lo que temo
es quedarme sin ellos.

Isaac Asimov (1920-1992) Escritor ruso.

Nowadays, technology is something that we use all the time in our daily life. As such, it is very important to make use of the resources it provides. In order to be able to use these tools, it is not enough to make them available to everyone. It's necessary to adapt and teach everyone to use them.

The main objective of this Final Degree Project is to complement the work of the colleagues who made the *Quizz* Android app last year, providing the teacher and its users with tools that allow them to manage the content of the app. For this, this work will focus on 2 fundamental aspects:

- The development of a web portal for students and teachers. This will allow students to send content to add to the *Quizz* app that, after the subsequent approval of their teacher, can become in games of this. On the other hand, teachers will be able to manage requests for the addition of content by their students and see statistics on the use of the app in order to guide or help them to improve in what they have more problems.

- The development of a parser that allows students, through a Word document with a specific format, to upload exercises of the different games. These Word documents should be sent to the professor who, after making the corrections that he considers appropriate, will upload to the web portal, where he will add the exercises to the app.

Keywords

Las computadoras son inútiles. Sólo pueden darte respuestas

Pablo Picasso, pintor

- Web App
- Parser
- Android
- Learning
- Game
- Content Management System

Índice

Índice de figuras

Capítulo 1

Introducción

La mejor forma de predecir el futuro es implementarlo

David Heinemeier Hansson, creador de Ruby on Rails

La relación entre la tecnología y cada aspecto de nuestra vida diaria se ha ido estrechando cada vez más en los últimos años. Actualmente nos es imposible entender ciertos aspectos sin la ayuda de ciertas herramientas, que nos hacen la vida más fácil.

La enseñanza por supuesto es otro de los aspectos que pueden y deben verse beneficiados por el uso de la tecnología, con nuevos métodos y herramientas que faciliten la labor, tanto de los docentes para enseñar como de los alumnos para aprender.

Dentro de este tipo de herramientas se encuentra *Quizz*, que es una aplicación móvil para Android desarrollada el curso pasado por dos alumnos de la facultad de informática de la Universidad Complutense de Madrid.

Quizz fue desarrollada en colaboración con estudiantes y profesor del *Grado en Estudios Ingleses de la Facultad de Filosofía y Letras de la Universidad Autónoma de Madrid*, y su propósito es ayudar a mejorar los resultados en exámenes.

Quizz se encuentra disponible en la *Google Play Store* desde el siguiente enlace:

`https://play.google.com/store/apps/details?id=es.quizz.quizz`

Esta aplicación tenía como aspecto negativo la dificultad para añadir contenido nuevo a la misma. Debido a esto nuestro director, el Dr. Gonzalo Méndez Pozo, nos propuso continuar el trabajo donde nuestros compañeros lo dejaron. Para ello, nos planteó varias alternativas:

- Desarrollar una aplicación propia para el profesor, que le permitiese gestionar los contenidos incluidos en la aplicación Android, así como

llevar un seguimiento del trabajo que hiciesen sus alumnos.

- Dado que los usuarios veían el documento Word como una forma cómoda para entregar contenido para ser usado por la aplicación, pensamos que era buena idea idea la de incluir un parseador de documentos Word. Este parseador sería el encargado de introducir nuevo contenido en la aplicación previa aprobación del mismo por parte del profesor.

- Añadir nuevos ejercicios a la aplicación Android, que mejoren la experiencia de los usuarios. Entre otras cosas, inicialmente pensamos en juegos como una *sopa de letras*.

- Implementar un mejor sistema de estadísticas para la aplicación Android, que registre los datos de uso de los alumnos y que pueda visualizar correctamente el profesor desde el portal web, de cara a poder orientar a los mismos y mandarles tareas para mejorar aquello en lo que vayan más justos.

De estas recomendaciones de nuestro director, decidimos que eran más necesarias aquellas que permitiesen añadir contenido a la aplicación; por lo que dimos prioridad a los desarrollos de la aplicación del profesor -añadiendo también un módulo para los alumnos- y el parseador. Después, si el tiempo nos lo permitía, abordaríamos el sistema de estadísticas y la inclusión de nuevos ejercicios.

Estructura de la memoria

A continuación se explica brevemente la estructura de la memoria:

Capítulo 1. Introducción

En este capítulo se incluyen la motivación y los objetivos que queremos cumplir con este trabajo.

Capítulo 2. Introduction

Este capítulo es la traducción al inglés del anterior.

Capítulo 3. Trabajo relacionado

En este capítulo se realiza una exposición de otros trabajos relacionados con este que nos ocupa, detallando las diferencias y similitudes, así como nuestra aportación al estado del arte.

Capítulo 4. Punto de partida

En este capítulo se explica más pormenorizadamente en qué consiste la aplicación Quizz, desarrollada el curso pasado en Android, y que es el punto de partida para nuestro trabajo.

Capítulo 5. Tecnologías utilizadas

En este capítulo comentamos brevemente las herramientas que hemos usado para el desarrollo de este proyecto, explicando para qué se ha utilizado cada una.

Capítulo 6. Portal web

En este capítulo se explica el procedimiento que se ha llevado a cabo y las tecnologías que se han utilizado para la elaboración del portal web.

Capítulo 7. Parseador de documentos Word

En este capítulo se profundiza en las técnicas que se han utilizado para el desarrollo del parseador de documentos Word, así como el formato específico que deben tener los mismos.

Capítulo 8. Estructura de la base de datos

En capítulo se detallan los aspectos técnicos más importantes de la aplicación, de cara a futuras mejoras y ampliaciones.

Capítulo 9. Conclusiones y trabajo futuro

En capítulo se exponen las conclusiones a las que se han llegado tras la realización de este trabajo, así como los aspectos que se deberían tener en cuenta en el futuro, si se quiere continuar con el mismo.

Capítulo 10. Conclusions and future work

Este capítulo es la traducción al inglés del anterior.

Trabajo individual

En este capítulo explicamos la aportación de cada uno de los integrantes a este trabajo.

Bibliografía

Por último, enumeramos todas las referencias bibliográficas que hemos tomado a lo largo de la elaboración de este trabajo incluyendo libros, artículos, páginas web de interés u otros trabajos.

Capítulo 2

Introduction

El logro más impresionante de la industria del software es su continua anulación de los constantes y asombrosos logros de la industria del hardware

Henry Petroski, especialista en análisis de fallos

The relationship between technology and every aspect of our daily life has been narrowed in the last few years. Currently it is impossible for us to understand certain aspects of our life without the help of certain tools, which make our lives easier.

Teaching is another aspect that can and should benefit from use of technology, with new methods and tools that facilitate the work, from teachers until students.

Within this type of tools *Quizz*, which is an Android mobile app developed last year by two students of the Computer Science School of Complutense University of Madrid.

Quizz was developed in collaboration with students and professor of *the Degree in English Studies of the Faculty of Philosophy and Philology of the Autonomous University of Madrid*, and its purpose is to help improve the results in exams. *Quizz* is available in *Google Play Store* at the following link:

`https://play.google.com/store/apps/details?id=en.quizz.quizz`

This application had as a negative aspect the difficulty to add new contents to it. Due to this, our director, Dr. Gonzalo Méndez Pozo, proposed us to continue the work where our colleagues left it. For this, he proposed several alternatives:

- Develop an application for the teachers that allow them to manage the contents included in the Android application, as well as keep track of

the work their students do.

- Since users viewed the Word document as a convenient way to deliver content for use by the application, we thought it was a good idea to include a Word document parser. This parser would be responsible for introducing new content in the application after its approval and correction by the teacher. The specific format will be explained later.

- Add new exercises to the Android application, which improve the user experience. Among other things, we initially thought of games as a *letter soup*.

- Implement a better statistics system for the Android application, which registers the student's use data and can be viewed correctly by the teacher from the web portal, in order to guide them and send them tasks to improve their results.

From these recommendations of our director, we decided that those that allowed adding content to the application were more necessary; so we gave more priority to the developments of the teacher's application - adding a module for the students - and the parser. Then, if time allowed, we would approach the statistics system and the inclusion of new exercises.

Document's structure

The structure of the memory is briefly explained below:

Chapter 1. Introducción

This chapter includes the motivation and objectives that we want to fulfill with this work.

Chapter 2. Introduction

This chapter is the English translation of the previous one.

Chapter 3. Related work

In this chapter an exhibition of other works related to this one is presented, detailing the differences and similarities, as well as our contribution to the state of the art.

Chapter 4. Starting point

In this chapter we explain in more detail what the Quizz application consists of, developed last year in Android, and that it is the starting point for our work.

Chapter 5. Technologies used

In this section we briefly comment on the tools we have used for the development of this project, explaining why each one has been used.

Chapter 6. Web Portal

This section explains the procedure that has been carried out and the technologies that have been used to prepare the web portal.

Chapter 7. Word document parser

This chapter delves into the techniques that have been used for the development of the Word document parser, as well as the specific format that they should have.

Chapter 8. Database structure

The chapter details the most important technical aspects of the application, for future improvements and extensions.

Chapter 9. Conclusiones y trabajo futuro

The chapter presents the conclusions that have been reached after the completion of this work, as well as the aspects that should be taken into account in the future, if someone wants to continue it.

Chapter 10. Conclusions and future work

This chapter is the English translation of the previous one.

Individual work

In this chapter we explain the contribution of each member to this work.

Bibliography

Finally, we list all the bibliographic references that we have taken throughout the preparation of this work, including books, articles, websites of interest or other works.

Capítulo 3

Trabajo relacionado

Las únicas personas que tienen algo que temer de software libre son aquellos cuyos productos tienen un valor aún menor

David Emery

En este capítulo queremos hacer un análisis de las aplicaciones que existen actualmente que estén relacionadas, de un modo u otro, con lo que hemos desarrollado en este trabajo. Para ello, hemos querido considerar tres aspectos fundamentales:

- Aplicaciones móviles o juegos educativos que permiten aprender mediante el uso de pruebas que ayudan al aprendizaje.
- Sistemas de gestión de contenidos CMS que permiten añadir, eliminar o modificar elementos en aplicaciones o páginas web.
- Parseadores de distinta índole, que permiten convertir un texto de entrada en otra estructura que entienda el programa que va a utilizar esa información.

3.1. Aplicaciones móviles educativas

Existen multitud de aplicaciones móviles cuya intención es, mediante diversas pruebas y juegos, ayudar al aprendizaje de una determinada materia o idioma. Entre ellas, podemos encontramos los siguientes.

3.1.1. Duolingo

Duolingo es una aplicación destinada al aprendizaje de idiomas. Cuenta con versión web, versión para (Xataka, 2014) y aplicaciones móviles Android,

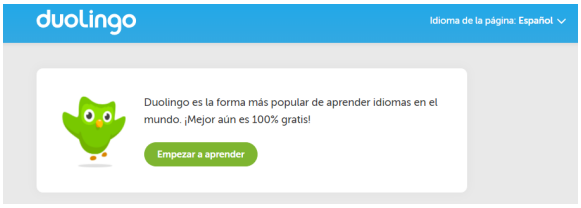


Figura 3.1: Captura de la aplicación Web Duolingo

iOS y Windows Phone. Se pueden aprender entre otros inglés, francés, alemán, portugués o italiano. Lo podemos ver en 3.1

Duolingo tiene como principal característica lo intuitiva y práctica que es. No es necesario leer textos en la mayoría de ocasiones, puesto que sólo con ver las imágenes se pueden llegar a entender el significado del mayor número de oraciones. Los progresos del usuario se ven por medio de gráficos que indican el grado de dominio del aspecto en concreto. Como aspecto curioso de esta aplicación, se incluyó la posibilidad de aprender el idioma *Valirio* de la serie *Juego de Tronos* (Xataka, 2017)

3.1.2. Google Classroom

Google Classroom es un servicio web gratuito que puede utilizar cualquier usuario que disponga de una cuenta personal de Google, o gestionada por Google. Está destinada a que alumnos y profesores se comuniquen fácilmente independientemente de donde se encuentren.

Con *Classroom*, los profesores pueden crear tareas, debatir sobre cualquier asunto relacionado con la materia o enviar notificaciones a los alumnos. Los contenidos están bien organizados por secciones y se pueden acceder a las tareas a realizar organizadas en el calendario de que dispone. Además, está perfectamente integrado con otras aplicaciones de Google como Documentos y Formularios de Google, Calendar, Gmail y Drive.

Classroom se puede acceder desde cualquier navegador y también dispone de aplicaciones móviles para Android e iOS.

3.1.3. Flashcards

Flashcards es una aplicación móvil para Android disponible en la *Google Play Store* que permite estudiar con la ayuda de Flashcards (fichas de aprendizaje). Cualquiera puede crearlas o hacer uso de las ya existentes. Está desarrollada por *Dictionary.com*. Podemos ver una pantalla de esta aplicación en la figura 3.2

Figura 3.2: Captura de la aplicación móvil Flashcards

3.1.4. Busuu

Busuu es una aplicación móvil que utiliza una metodología de red social para ayudar al aprendizaje de idiomas. Tiene a muchos usuarios registrados y da la posibilidad de aprender nueve idiomas. Se agrupa por niveles y etapas, de forma que primero pretende que se aprendan las palabras y su pronunciación, después éstas son incluidas en textos para su comprensión y por último se pretende que sean los propios usuarios quienes escriban textos legibles. Estos textos será enviados para que otro usuario los corrija o aconseje sobre los mismos.

3.2. Sistemas de gestión de contenidos web

A la hora de gestionar un proyecto online, es importante contar con un gestor de contenido que facilite la inclusión, modificación o eliminación de material en las aplicaciones, ya sean de escritorio, web o móviles. Entre los gestores de contenido existentes, encontramos los siguientes:

3.2.1. WordPress

WordPress es un gestor de contenidos muy popular. Dispone de miles de usuarios y tiene una interfaz fácil de utilizar. Incluye plantillas para facilitar la inserción de material. Tiene una enorme comunidad de desarrolladores y diseñadores que lo han programado o creado complementos y plantillas para la comunidad. Algunas de sus características son las siguientes:

- Sencillez a la hora de instalar, modificar y personalizar.
- Publica las entradas ordenadas por fecha.
- El diseño visual depende de las plantillas utilizadas.
- Separa el contenido del diseño en XHTML (Wikipedia, XHTML) y CSS (Wikipedia, CSS).
- Publicación mediante correo electrónico.

3.2.2. Drupal

Drupal es un CMS libre, con licencia GNU/GLP (GNU, 2018). Tiene un diseño adecuado a construir y llevar la gestión de comunidades en internet. Tiene una gran cantidad de módulos, lo que le da una gran versatilidad a la hora de crear sitios web muy diversos.

Entre las aplicaciones que pueden hacer uso de sus servicios, podemos encontrar las siguientes:

- Webs corportativas.
- Blogs.
- Aplicaciones de comercio electrónico.
- Portales web de periódicos.

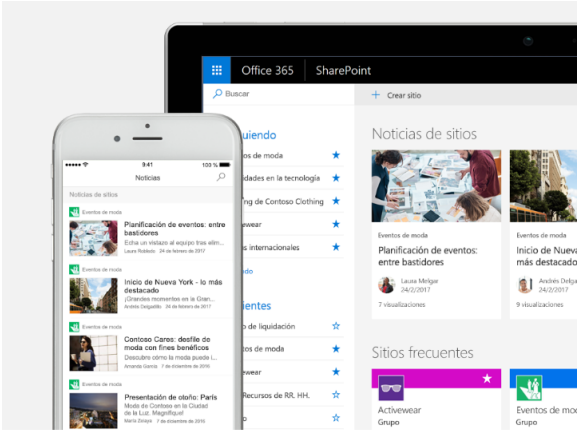


Figura 3.3: Aplicación de SharePoint para móvil y Tablet

3.2.3. SharePoint

SharePoint es una plataforma de colaboración empresarial que incluye diferentes servicios para administración de procesos, módulos de búsqueda o para manejar documentos fácilmente. Lo vemos en la Figura 3.3

Puede ser utilizado en sitios web para acceder a espacios de trabajo compartidos. Microsoft fue su desarrollador y lo integran las siguientes tecnologías, entre otras:

- *SharePoint services*, que ayuda a compartir y administrar documentos en una web.
- *Search Server*, que permite buscar cualquier información que necesitemos de manera rápida y eficaz.
- *Forms Server*, que permite convertir formularios de *InfoPath* (Wikipedia, InfoPath) en *html* o *ajax*.

3.3. Parseadores de documentos

La mayoría de parseadores que podemos encontrar se utilizan para parsear archivos en formato JSON (web docs, 2017) o XML (Linux, 2017), pero es muy complicado encontrar alguno que parsee documentos Word a no ser que éstos posean la estructura de un XML o un JSON. Algunos de los que hemos encontrado son los siguientes:

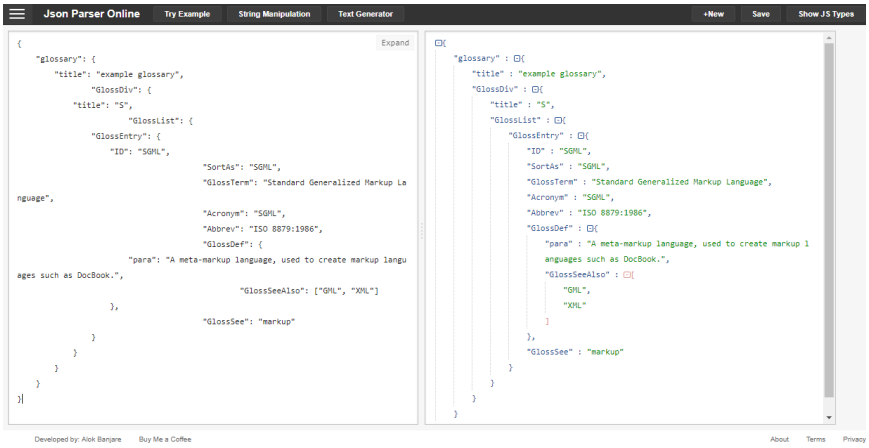


Figura 3.4: Captura del portal Json Parser Online

3.3.1. Json Parser Online

Se trata de un portal web que realiza parseos en tiempo real de textos, convirtiéndolos a formato JSON y permitiendo guardarlos posteriormente. Este portal informa en tiempo real si el formato o la estructura no son los correctos. Además de ello, se pueden ver los tipos de los diferentes elementos que integran el documento JSON resultante o enumerar los elementos que integren arrays dentro.

3.3.2. XML Formatter

XML Formatter es una herramienta online que permite formatear un texto que posea la estructura de un documento XML para que su lectura mejore, dejándolo con una indentación adecuada.

3.3.3. PEG.js

PEG.js es un parseador para *JavaScript* basado en expresiones de análisis gramatical. Permite, por ejemplo, introducir operaciones aritméticas sencillas que, tras analizarlas y traducirlas, devuelven un resultado.

3.4. Trabajo previo: Quizz

3.4.1. Descripción, estructura y diseño

Quizz es una aplicación cuyo objetivo es mejorar los resultados en exámenes. Inicialmente se desarrolló en colaboración con alumnos del Grado en Estudios Ingleses de la Facultad de Filosofía y Letras de la Universidad

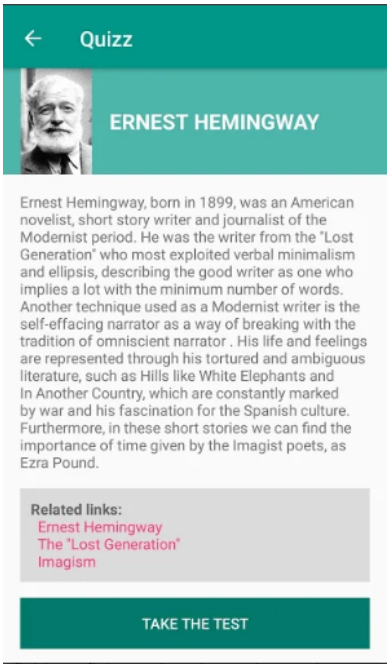


Figura 3.5: Información de un autor en Quizz

Autónoma de Madrid y su profesora, la Dra. Ana González-Rivas Fernández (Ángela Rocío Camargo Sánchez y Fernández, 2017).

Esta tarea la lleva a cabo por el medio de una serie de juegos y pruebas muy diversas. Se trata de la aplicación de la cual parte nuestro trabajo. Los materiales que se gestionan mediante nuestro trabajo tendrán como objetivo final mejorar y ampliar los contenidos en esta aplicación.

Quizz tiene un diseño Material Design (Wikipedia, Material Design) muy cuidado y una interfaz muy intuitiva, que facilita mucho las cosas para sus usuarios.

Para entender el funcionamiento de la aplicación web, vamos a proceder a explicar primero la aplicación Android y como está estructurada esta, para que conceptos como "contenidos" o "secciones" no nos sean extraños.

Al entrar a la aplicación, lo primero que vemos es un formulario de login o registro donde el usuario deberá loguearse o registrarse para poder acceder a las funcionalidades de la aplicación.

Una vez logueado, al usuario se le muestra una lista de las clases a las que tiene acceso, para poder acceder a ellas. Es necesario introducir un código, proporcionado por el profesor, que está asociado a dicha clase.

Al acceder a una clase, se muestran las distintas secciones de esta, una clase puede tener varias secciones.

Al acceder a una sección, se muestran los distintos materiales de ésta,

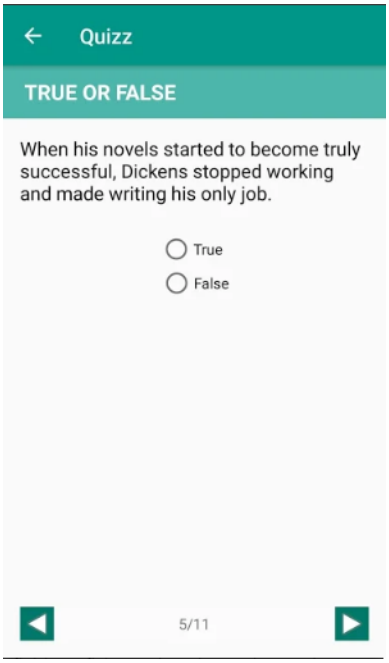


Figura 3.6: Ejemplo de ejercicio en Quizz

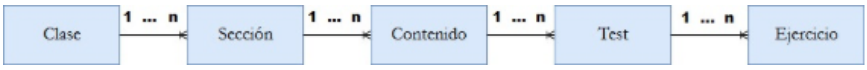


Figura 3.7: Estructura de la aplicación Android

porque una sección puede tener varios contenidos. Para el caso de la posible sección "Siglo XIX en literatura Inglesa" un contenido posible sería "Robert Browning", como podemos ver en la figura 3.5

Una vez seleccionamos el contenido, accedemos a una descripción de este, con una imagen y un link para mayor información, también la posibilidad de realizar un test aleatorio, el cual tiene varios ejercicios relacionados con dicho material.

Podemos ver una estructura simplificada de esto en la Figura 3.7

3.4.2. Tipos de ejercicios

- En el primer ejercicio, *True or False*, se presenta un enunciado al usuario y este tiene que decidir si es correcto o no lo que se expone en él. Lo vemos en la figura 3.6.
- En *Multiple Answer* el usuario, después de leer el enunciado de la pregunta, tiene que escoger entre varias opciones cuál o cuáles de ellas

responden a la pregunta formulada.

- En *Fill in the Gap* el usuario deberá leer un texto incompleto que debe completar haciendo uso de las diferentes opciones que se le presentan.
- *Pair Words* es un tipo de ejercicio en el que el usuario debe relacionar conceptos afines entre ellos teniendo en cuenta el enunciado de la pregunta.
- En el tipo de ejercicio *Direct Question*, el usuario debe responder a una cuestión concreta escribiendo la respuesta considere correcta para la resolución del ejercicio.
- Respecto al ejercicio *Order Sentences*, el usuario tendrá que leer diferentes oraciones que se encuentran dispuestas en orden incorrecto y proceder a ordenarlas correctamente teniendo en cuenta el enunciado del ejercicio.
- Por último, el ejercicio *Recognise the text* presenta un fragmento de texto, para que el usuario seleccione qué título y tema le corresponden.

3.4.3. Modelo de datos

La aplicación Android de *Quizz* usaba una base de datos *SQL* con una cantidad considerable de tablas, como podemos ver en la Figura **??**

A pesar de estar desarrollada principalmente para la clase de Filología inglesa, y para el uso de la aplicación por parte de alumnos, la base de datos está estructurada para que sea posible extender su funcionalidad a varias clases y asignaturas, y para añadir nuevos tipos de ejercicios. También es posible contar con varios tipos de usuario, aunque en la aplicación no se use este dato.

3.4.4. Conclusión

Quizz es una aplicación eficiente, fácil de usar y agradable a la vista, extensible a varias asignaturas y se pueden añadir una infinidad de tipos de ejercicio nuevos. Su principal problema es que, para añadir contenidos, se tiene que hacer directamente en la base de datos. Esto impide que los profesores actualicen sus clases, y que nuevos profesores hagan uso de la aplicación. El profesor tampoco dispone de ninguna herramienta para ver qué alumnos hay en su clase o si usan la aplicación. Por lo tanto, decidimos que desarrollar un sistema de subida de contenido y de visualización de datos para el profesor, era una prioridad por encima de la mejora de la aplicación Android.

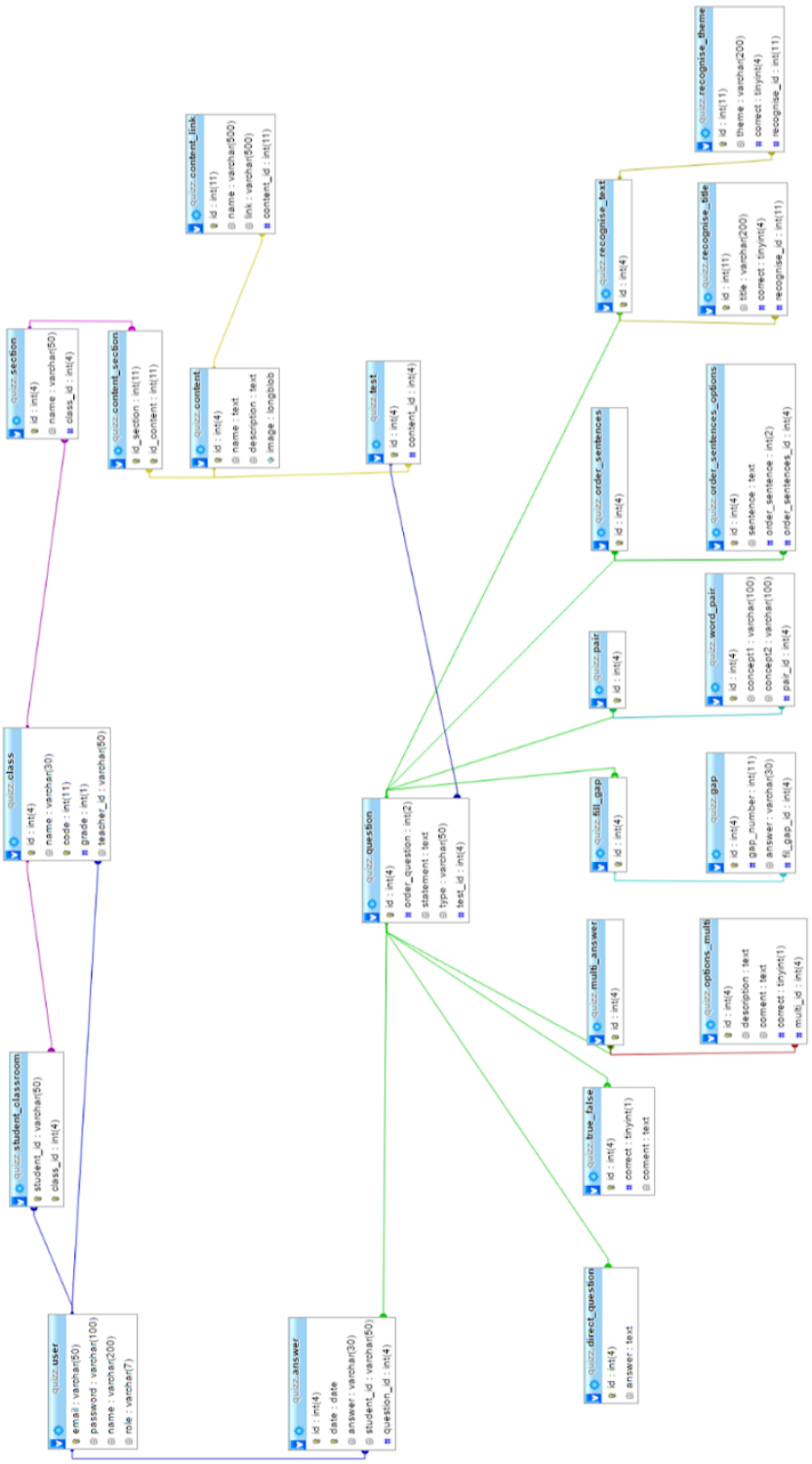


Figura 3.8: Modelo de datos de la aplicación

Capítulo 4

Punto de partida

La informática tiene que ver con los ordenadores lo mismo que la astronomía con los telescopios.

Edsger W. Dijkstra (1930-2002)
científico de la computación holandés.

Nuestro punto de partida fue la aplicación Android desarrollada por otros alumnos de la UCM como trabajo de fin de grado el año anterior Ángela Rocío Camargo Sánchez y Fernández (2017), la cual permitía que los alumnos realizasen diferentes tipos de ejercicios y accediesen a distintos tipos de contenidos. La aplicación se centraba principalmente en la enseñanza de literatura inglesa, pero fue diseñada de tal forma que permitiese la adaptabilidad y expansión a otras asignaturas. De igual manera, en cuanto a funcionalidad, solo tenía un tipo de usuario, pero estaba pensada para poder diferenciar el tipo entre estudiantes y profesores.

Al comenzar nos dimos cuenta de que el mayor punto débil que tenía la aplicación es que los profesores y alumnos no podían subir sus propios contenidos, sino que tenían que enviarlos en un Word a los desarrolladores de la aplicación, y estos los insertaban directamente en la base de datos, de tal forma, el profesor no tenía ningún tipo de acceso a la información de sus alumnos ni al contenido de sus clases, tampoco podía crear nuevas clases, esto hacía que el añadir contenido a la aplicación fuese muy difícil y tedioso.

Teniendo en cuenta esto, decidimos enfocarnos principalmente en estas carencias, para que tanto el profesor como los alumnos pudieran añadir contenidos y ejercicios de forma sencilla, para esto, sopesamos dos opciones:

- La primera era que los alumnos rellenasen un Word, con un determinado formato, y luego lo subiesen a la aplicación, y este Word se parseara automáticamente y añadiese los contenidos a la base de datos. Esta forma era la usada en la primera versión de *Quizz* (Porta, 2016), mediante

una plataforma web muy sencilla, pero no se llegó a continuar con el desarrollo de esta y quedó obsoleta.

- La segunda era tener una serie de formularios web que permitiesen añadir contenidos y ejercicios de forma sencilla e intuitiva.

Como no sabíamos cuál era el más adecuado y cuál preferirían profesor y alumnos, decidimos desarrollar los dos métodos en paralelo, para posteriormente, realizar un estudio sobre qué método era el más indicado.

El objetivo era realizar ambos desarrollos de tal manera que fuesen fácilmente extensibles a cualquier tipo de asignatura teórica, y así intentar llevar *Quizz* al máximo número de asignaturas posibles.

De tal manera, para dar acceso a los usuarios a estos métodos de subida, se decidió crear una plataforma web, para profesor y alumnos, que permitiera a los alumnos subir contenido, el cual, posteriormente, el profesor pudiera aceptar o rechazar.

El profesor, podría ver toda la información y contenido de sus clases y alumnos, así como las solicitudes de contenido de estos, y también crear nuevas clases y contenido.

Con esta aplicación web también se pretende que el profesor tenga acceso a una serie de estadísticas de los alumnos, como pueden ser ejercicios realizados, contenido aportado, nota de los test realizados etc... Y también estadísticas sobre el contenido y los ejercicios de una determinada clase, para ver cuáles son los que reciben mejores notas por parte de los alumnos.

Esta aplicación distinguirá entre alumnos y profesores, y, dependiendo del tipo de usuario, mostrará distintas funcionalidades.

Permitirá que teniendo una cuenta de tipo profesor, se creen clases, las cuales tendrán un código predefinido y a partir de este código, desde la aplicación web, los alumnos se podrán registrar en dicha clase. De esta forma se automatiza el proceso de creación de clases y contenidos, y lo único que tendría que hacer un profesor para utilizar nuestra aplicación sería solicitarnos una cuenta.

En cuanto a la parte de Android, el objetivo inicial era añadir nuevos tipos de ejercicio, pero decidimos darle prioridad a añadir contenido y a mostrar estadísticas al profesor, para lo cual debíamos modificar la aplicación Android para guardar los datos necesarios cuando los alumnos realizasen un ejercicio.

Es importante recalcar que la aplicación web está hecha de tal forma que un profesor solo pueda acceder a la información de sus propios alumnos y a los contenidos de sus propias clases, así como los alumnos solo pueden añadir contenido a clases en las que estén registrados. De esta forma por muchas clases y profesores que haya en la aplicación, cada uno trabajará en el contexto de sus clases sin poder acceder a información ajena a éstas.

Para que un alumno se registre en una clase, debe recibir el código de

la clase por parte del profesor, y registrarse en la clase introduciendo dicho código en la aplicación Android.

Es importante destacar que aparte de realizar una aplicación web, que responda a una serie de necesidades reales, también pretendemos usar y adquirir conocimientos de tantas tecnologías web como podamos.

Capítulo 5

Tecnologías utilizadas

No documentes el problema; arréglalo
Atli Björgvin Oddsson, programador

5.1. Lenguajes de programación

5.1.1. Javascript

Javascript es un lenguaje de programación interpretado orientado a objetos y débilmente tipado. Se utiliza principalmente en su forma del lado del cliente (Wikipedia, ClientSide), implementado como parte de un navegador web. Gracias a tecnologías como *Node.js* es posible usar *Javascript* en el lado del servidor, lo cual hemos hecho en este proyecto.

Todos los navegadores modernos interpretan el código *Javascript* integrado en las páginas web. Para interactuar con una página web se provee al lenguaje *Javascript* de una implementación del árbol *DOM* (Wikipedia, DOM).

Javascript es uno de los lenguajes de programación más usados del mundo, sobre todo para el desarrollo de aplicaciones web. Hay una gran cantidad de tecnologías desarrolladas para *Javascript*, de código libre, que facilitan el desarrollo y permiten crear aplicaciones web de manera más eficiente. Éstas han sido las razones por las que nos hemos decantado por *Javascript* para el desarrollo tanto de la aplicación web como del parseador de documentos.

5.1.2. SQL

SQL es un lenguaje declarativo estándar internacional de comunicación dentro de las bases de datos que permite el acceso y manipulación de datos en una base de datos relacional. El alcance de *SQL* incluye la inserción de datos, consultas, actualizaciones y borrado, la creación y modificación de esquemas y el control de acceso a los datos.

5.1.3. HTML5

HTML5 (HyperText Markup Language, versión 5) (Wikipedia, HTML5) es un lenguaje de marcado para estructurar y presentar el contenido para la web. Es el más usado con diferencia. A día de hoy, la gran mayoría de las páginas de web usan este lenguaje y todos los navegadores son compatibles con él.

HTML5 contiene grandes ventajas respecto a sus versiones anteriores como una nueva estructura de etiquetas mejorada, inclusión de las etiquetas vídeo y audio o capacidad de realizar ejecuciones offline de las páginas web creadas con código *HTML5*. Incluye una nueva etiqueta de dibujo sobre la página web, llamada *canvas*, que vuelve el proceso de crear dibujos en el sitio web tan fácil como dibujar con aplicaciones como *Paint*. Ofrece la posibilidad de obtener un código más limpio y fácil de depurar, que los códigos de los estándares anteriores.

5.1.4. CSS

CSS (Cascading Stylesheets) (Wikipedia, CSS) es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Es muy usado para establecer el diseño visual de páginas web e interfaces de usuario escritas en HTML.

Como ya mencionamos, *CSS3* sirve para cambiar el aspecto de un sitio web, desde las medidas para los márgenes hasta las especificaciones para las imágenes y el texto. *CSS3* funciona mediante módulos, algunos de los más comunes son "colors", "fonts", "backgrounds", etc. Los módulos son sólo categorías en las que se pueden dividir las modificaciones que hacemos al aspecto de nuestro sitio web. Existen un gran variedad de módulos los cuales nos permiten desde manejar fuentes hasta manejar animaciones.

5.2. Entornos de desarrollo y tecnologías usadas para el desarrollo

5.2.1. Visual Studio Code

Visual Studio Code es un editor de código fuente y un entorno de desarrollo Integrado desarrollado por Microsoft, es compatible con gran cantidad de lenguajes de programación, gratuito y de código abierto y multiplataforma (Windows, Linux y MAC). Las características principales de Visual studio code son:

- Cuenta con un avanzado depurador de código que permite conocer todos los problemas y errores en tiempo real, ayudando a identificar y

resolverlos.

- Permite desde resaltar la sintaxis de nuestros proyectos hasta hacer uso de auto-completar funciones, controlar nuestras variables y, además, ver definiciones de las funciones de los distintos lenguajes de programación.
- Se integra en *Git*, que permite llevar un control de versiones de forma sencilla desde esta plataforma.
- Desde el mismo editor vamos a poder acceder a una gran variedad de extensiones y complementos que nos va a permitir añadir las funciones y características que necesitemos.

5.2.2. XAMPP

XAMPP es un paquete de software libre (que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl).

XAMPP te permite instalar de forma sencilla *Apache* en tu propio ordenador, sin importar tu sistema operativo incluye además servidores de bases de datos como *MySQL* y *SQLite* con sus respectivos gestores *phpMyAdmin* y *phpSQLiteAdmin*.

Con *XAMPP* es posible levantar un servidor en *localhost* usando *Apache* sin necesidad de tener acceso a internet, lo cual nos ha permitido desarrollar y probar nuestro proyecto en *localhost*.

XAMPP también contiene la herramienta *phpMyAdmin*, la cual hemos usado para explorar, modificar y ampliar la base de datos existente.

phpMyAdmin es una herramienta escrita en *PHP* que permite administrar bases de datos *SQL* a través de un portal web, puede crear y eliminar bases de datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia *SQL*, administrar claves en campos, administrar privilegios y exportar datos en varios formatos.

5.2.3. NetBeans

NetBeans es un IDE (Wikipedia, IDE) libre, gratuito y sin restricciones de uso. Posee gran cantidad de módulos que permiten usarlo para programar en gran cantidad de lenguajes. En este trabajo lo hemos utilizado en la parte del Parseador, por su posibilidad de utilizarlo para programar en *Javascript*.

5.2.4. Google Drive

Google Drive es un servicio de alojamiento de archivos propiedad de Google. Lo hemos usado para intercambiar archivos entre nosotros y con

el profesor, tanto del proyecto de la aplicación Android de años anteriores como los del proyecto del año en curso y para redactar y editar partes de la memoria de manera conjunta antes de su incorporación a LaTeX.

5.2.5. Github

GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones *Git*. *GitHub* aloja tu repositorio de código y te brinda herramientas muy útiles para el trabajo en equipo, dentro de un proyecto. Además de eso, puedes contribuir a mejorar el software de los demás. Para poder alcanzar esta meta, *GitHub* provee de funcionalidades que permiten contribuir en proyectos ajenos controlando el acoplamiento del código.

Hemos usado *GitHub* para compartir las distintas versiones del código de nuestra aplicación a medida que lo íbamos desarrollando y para descargar el código de la aplicación Android.

5.3. Tecnologías web

Node.js

Node.js es un entorno de ejecución para *Javascript* construido con el motor de *Javascript V8* del navegador *Google Chrome*. *Node.js* usa un modelo de operaciones E/S sin bloqueo y orientado a eventos, que lo hace liviano y eficiente. El ecosistema de paquetes de *Node.js*, *npm*, es el ecosistema más grande de librerías de código abierto en el mundo.

Node.js incorpora varios "módulos básicos", pero su punto fuerte es que puede incorporar módulos de terceros con facilidad mediante el comando "npm install", esto nos permite usar una gran cantidad de funcionalidades desarrolladas por terceros en nuestro proyecto, y asegurándonos de que todas estas son de código libre. Estos paquetes son completamente reutilizables.

De *Node.js* cabe destacar que tiene un ecosistema y comunidad de desarrolladores de terceros muy activa, con cantidad de gente deseosa de ayudar.

Express.js

Express.js es el framework web más popular de *Node*, y es la librería subyacente para un gran número de otros *frameworks* web de *Node* populares.

Express.js, según sus creadores, es un framework de desarrollo de aplicaciones web minimalista y flexible para *Node.js*. Está inspirado en *Sinatra*, y es una extensión de *Connect* además es robusto, rápido, flexible y muy simple.

Proporciona mecanismos para:

- Facilidades para enrutamientos de URL basados en el protocolo http (*Get*, *Post*, *Put*...)
- Integración con motores de renderización de "vistas" para generar respuestas mediante la introducción de datos en plantillas.
- Facilidades para motores de plantillas como *Jade*, *EJS* o *JinJS*.
- Establecer ajustes de aplicaciones web como qué puerto de la conexión, y la localización de las plantillas que se utilizan para renderizar la respuesta.
- Añadir procesamiento de peticiones "middleware" adicional en cualquier punto dentro del manejo de la petición.

A pesar de que *Express* es en sí mismo bastante minimalista, los desarrolladores han creado paquetes de middleware compatibles para abordar casi cualquier problema de desarrollo web. Hay librerías para trabajar con *cookies*, sesiones, inicios de sesión de usuario, parámetros URL, datos POST, cabeceras de seguridad y muchos más. Puedes encontrar una lista de paquetes *middleware* mantenida por el equipo de *Express.js* en *Express Middleware* (junto con una lista de algunos de los paquetes más populares de terceros).

Para instalar *Express.js* desde node, solamente hay que ejecutar el siguiente comando:

```
1 npm install express --save
```

Para usar express, tenemos que incluir las siguientes sentencias en nuestro fichero principal, app.js:

```
1 const express = require("express");
2 const app = express();
```

El módulo express exporta una única función. Cada llamada a esta función devuelve una aplicación. Una aplicación es un servidor HTTP que escucha en un determinado puerto. A continuación se definen los manejadores de ruta, que especifican las acciones del servidor para cada URL:

```
1 app.get(URL, callback)
```

Cuando se reciba una petición de tipo GET sobre la URL pasada como parámetro, se llamará a la función callback, que será la que gestione la petición.

5.3.1. EJS

EJS es un motor de plantillas sencillo de usar y depurar, además de ser muy rápido. Está realizado en *Javascript* y está muy aconsejado su uso en combinación de *NodeJS* y *Express*. Permite procesar paginas html (cuya

extensión será .ejs) en el servidor antes de servirlas al cliente, de forma que estas podrán tener partes dinámicas.

La manera recomendada de trabajar con él es usarlo como una dependencia de *NodeJS* en nuestros proyectos, para instalarlo solo hay que utilizar el comando:

```
npm install ejs --save
```

Para utilizar *EJS* como motor de plantillas en una aplicación *Express*, incluiremos la siguiente línea de código en nuestro fichero principal:

```
app.set("view engine", "ejs");
```

Las plantillas ejs son documentos html con marcadores especiales de distintos tipos:

- Marcadores de programa, delimitados por $<\%\%>$. Los marcadores de programa contienen sentencias (o fragmentos de sentencias) *Javascript*. Suelen utilizarse con bucles, condicionales, etc.

```
<% if (!usuario.nombre) { %>
  <p>No estás identificado.</p>
<% } else { %>
  <p>¡Bienvenido, <%= usuario.nombre %>!</p>
<% } %>
```

- Marcadores de expresión, delimitados por $<\%^{-}\%>$. Los marcadores de expresión evalúan la expresión *Javascript* dada, la convierten en cadena, y se reemplazan por dicha cadena.

```
<h1> ¡Bienvenido, <%= usuario.nombre %>!<h1/>
```

5.3.2. Express-session y express-mysql-session

Express-session es un *middleware* que gestiona el almacenamiento de sesiones. Este *middleware* añade un atributo session al objeto request que contiene los datos de sesión correspondientes al cliente que se está atendiendo.

El atributo *request.session* también se utiliza para añadir o modificar información de la sesión. Al utilizar *response.end()* o similar, el servidor guarda toda la información de *request.session* en el almacenamiento de sesiones del servidor.

Para instalarlo solo hay que utilizar el comando:

```
npm install express-session --save
```

El módulo *express-mysql-session* permite almacenar la información de sesión en una base de datos *MySQL*.

Para instalar este módulo se puede utilizar el comando:

```
npm install express-mysql-session --save
```

Para usar estos modulos debemos importar los modulos a nuestro fichero principal, crear un *sessionStore* con los datos de la base de datos y pasarle estos datos a nuestro *middleware* de *express session*, de esta forma tendríamos un sistema de sesiones que se alojan en una base de datos *MySQL*.

5.3.3. MySql

El paquete mysql nos permite realizar conexiones con una base de datos de una forma simple, facilitando el acceso y el manejo de estas.

Para instalarlo podemos usar el comando:

```
npm install mysql --save
```

Para usarlo debemos incluir en nuestro proyecto la siguiente sentencia:

```
var mysql = require("mysql");
```

Los pasos para realizar una conexión, utilizando este módulo son:

1. Crear e inicializar un objeto *Connection*.
2. Realizar la conexión.
3. Realizar la consulta o modificación.
4. Cerrar la conexión.

5.3.4. AdminLTE

AdminLTE es un panel de administración para *Bootstrap* creado por el estudio *Almsaeed*. Es una solución de código abierto basada en un diseño modular que permite una construcción y personalización sencillas. La idea es que cada uno de estos elementos sea un plugin o un widget a través del cual uno va creando la interfaz de usuario.

AdminLTE se puede descargar como un módulo ya compilado para su uso directo, o descargar su código fuente por si el desarrollador desea modificar sus componentes. *AdminLTE* está basado en *Bootstrap 3* y usa varios módulos node como *JQuery* o *Charts.js*.

5.3.5. Datatables

DataTables es un plugin para la librería de Javascript *JQuery*. Es una herramienta flexible y sencilla de usar que permite crear tablas *HTML* con funcionalidades como buscar, ordenar o paginar de una forma completamente dinámica y sin necesidad de recargar la página. Las funcionalidades también son customizables, pudiendo elegir cuales de estas usar.

Para usar este módulo se debe incluir en el fichero *HTML* el codigo *js* y *css* de *Datatables*, o bien descargándolo, o bien directamente desde la url proporcionada desde su web.

5.3.6. Bootstrap

Bootstrap es un framework desarrollado y liberado por *Twitter* que tiene como objetivo facilitar el diseño web. Permite crear de forma sencilla webs de diseño adaptable, es decir, que se ajusten a cualquier dispositivo y tamaño de pantalla y siempre se vean igual de bien. Es *Open Source* o código abierto, por lo que lo podemos usar de forma gratuita y sin restricciones.

Entre las ventajas de *Bootstrap* se encuentran:

- Un sistema de maquetado basado en columnas, lo cual hace que el diseño web sea sencillo y ordenado, y evita muchos de los problemas que se tenían al programar en css puro.
- Es *responsive*, por lo que se adapta fácilmente a cualquier dispositivo.
- Tiene una comunidad muy amplia y es de código libre por lo que hay una gran cantidad de plugins que pueden ser de utilidad.
- Se integra muy bien con las principales librerías *Javascript*.
- Permite utilizar muchos elementos web, combinando *HTML5*, *CSS* y *Javascript*.

Para utilizarlo en nuestro proyecto, solo hay que incluir sus ficheros en nuestro *HTML*, esto se puede hacer de varias maneras, descargándolo desde su web, usando un link directo al código o bien usando el gestor de paquetes npm, con el comando **$npm install bootstrap**. Para usar *Bootstrap* es necesario incluir también las librerías de *jQuery* y *Popper.js*.

5.3.7. JQuery

JQuery es una librería multiplataforma de *Javascript*, que nos permite simplificar el documento *HTML*, manejar eventos, animaciones y utilizar interacciones *AJAX* para el desarrollo web. *JQuery*, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en *JavaScript* que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

La característica principal de la biblioteca es que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol *DOM* y peticiones *AJAX*. Para ello utiliza las funciones $() o *JQuery()*.

Para incluirla en nuestro proyecto solo es necesario incluir la librería en nuestro documento *HTML*, descargandola desde su web o bien usando un link directo al código. Para incluirla en nuestro documento *HTML* solo hay que usar la siguiente línea de código:

```
1 <script src="pathToJs/js/jquery-3.2.1.min.js"> </script>
```

También es posible descargarla por medio del gestor de paquetes npm usando la instrucción: **$ npm install jquery**

Charts.js

Charts.js es una librería popular que permite crear gráficos elegantes y responsive sobre *Canvas HTML5*. Esta librería nos permite convertir datos puros a gráficas visualmente llamativas de una manera sencilla y rápida.

La librería te permite mezclar diferentes tipos de gráficos y trazar datos en escalas fecha tiempo, logarítmica, o personalizada con facilidad. También soporta animaciones que pueden ser aplicadas cuando se cambian los datos o se actualizan colores.

Se puede descargar la librería desde el gestor de paquetes npm por medio de la instrucción **$ npm install chart.js –save** o bien descargarla desde su web.

Para usar esta librería es necesario añadir la librería en nuestro documento *HTML*, podemos hacerlo con el siguiente código:

```
1 <script src="path/to/chartjs/dist/Chart.js"> </script>
```

Una vez añadida podremos crear gráficas desde nuestro archivo *js* de la siguiente manera:

```
1 <script> var myChart = new Chart(ctx, {...}); </script>
```

5.3.8. Socket.io

Socket.io es una librería en *JavaScript* para *Node.js* que permite una comunicación bidireccional en tiempo real entre cliente y servidor. Para ello se basa principalmente en *Websocket* pero también puede usar otras alternativas como sockets de *Adobe Flash, JSONP polling o long polling en AJAX*, seleccionando la mejor alternativa para el cliente justo en tiempo de ejecución. Esta librería es muy compatible con *Express.js* y es recomendable su uso con este.

Socket.io es realmente potente y podemos hacer todo tipo de aplicaciones en tiempo real, es muy útil, por ejemplo, para implementar chats en tiempo real. Esta librería no es tan sencilla de usar como las expuestas anteriormente, pero una vez se comprende cómo funciona puede ofrecer una gran cantidad de ventajas a la hora de desarrollar aplicaciones web en tiempo real.

Para descargarla, podemos usar el comando:

$ npm install socket.io

A la hora de usarla es necesario indicar a socket.io que escuche en el servidor de nuestra aplicación, esto lo podemos hacer usando el siguiente código:

```
1 var express = require('express');
2 var app = express();
3 var server = require('http').Server(app);
4 var io = require('socket.io')(server);
```

5.4. Otros

5.4.0.1. Latex

LaTeX es un sistema de composición de textos, orientado especialmente a la creación de libros, documentos científicos y técnicos que contengan fórmulas matemáticas. Se compone de una serie de macros que ayudan a usar el lenguaje TeX (Wikipedia, TeX).

Es muy utilizado para la composición de artículos académicos, tesis y libros técnicos, dado que la calidad tipográfica de los documentos realizados con LaTeX es comparable a la de una editorial científica de primera línea. LaTeX es software libre bajo licencia *LPPL*.

LaTeX presupone una filosofía de trabajo diferente a la de los procesadores de texto habituales y se basa en instrucciones, permite a quien escribe un documento centrarse exclusivamente en el contenido, sin tener que preocuparse de los detalles del formato. Además de sus capacidades gráficas para representar ecuaciones, fórmulas complicadas, notación científica e incluso musical, permite estructurar fácilmente el documento -con capítulos, secciones, notas, bibliografía o índices analíticos-, lo cual brinda comodidad y lo hace útil para artículos académicos y libros técnicos. LaTeX facilita la creación de documentos de un modo consistente mediante la separación del contenido de la presentación. Este mismo principio de separación del contenido y presentación/estilo es usado en la páginas web por *HTML* y *CSS*.

Capítulo 6

Portal web

Los estándares son siempre obsoletos.
Eso es lo que los hace estándares

Alan Bennett

6.1. Introducción

En este capítulo explicaremos todo el trabajo de diseño, implementación y planificación, realizado durante el desarrollo del portal web. Además, se podrán ver los cambios realizados debido a los contratiempos surgidos o las variaciones de los requisitos por parte de los clientes que han ocurrido durante el desarrollo del mismo.

El capítulo está dividido en varias secciones ordenadas cronológicamente, en las que se ve el progreso que ha tenido el proyecto. En cada una de ellas se detallan los procesos que se han realizado para conseguir los objetivos impuestos. Como esquema general de las funcionalidades de la aplicación web, podemos consultar el diagrama de casos de uso, el cual podemos encontrar en la figura 6.1.

6.2. Herramientas, entornos y estructura de la aplicación

Teniendo en cuenta que *Quizz* es un proyecto de larga duración, que seguramente será continuado por otros alumnos en años posteriores, hemos decidido que sería una buena idea explicar la estructura interna de la aplicación web, y la base de datos, para así facilitar la extensión de estas en un futuro y ahorrar trabajo a los alumnos que continúen el proyecto.

De esta forma también podemos explicar de una manera más clara y concisa el trabajo realizado, lo cual hará que esta memoria sea más completa. En

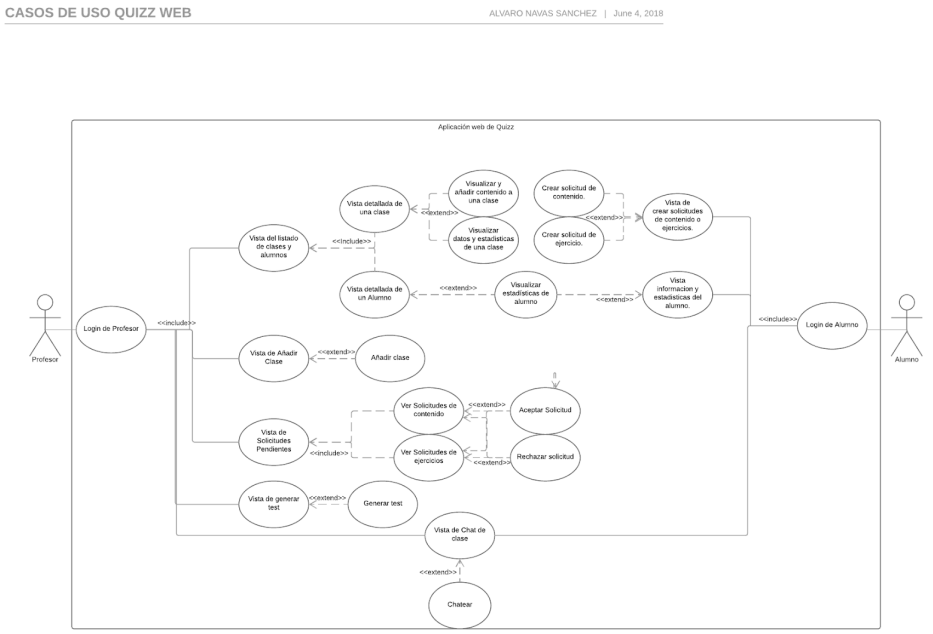


Figura 6.1: Diagrama de casos de uso de la aplicación

este apartado vamos a proceder a explicar cómo está implementada la aplicación web y su estructura interna, así como donde será necesario modificar el código para añadir funcionalidad a ésta.

Dado que teníamos conocimientos previos sobre desarrollo web en *Javascript*, decidimos usar este lenguaje para crear la aplicación web, usando el entorno *Node.js* por su eficiencia y gran versatilidad y con el framework *Express.js*, el cual hemos estudiado en algunas asignaturas de la carrera (ManuelMontenegro, 2018).

Para las vistas, se decidió usar el motor de plantillas *EJS*, el cual es altamente compatible con express y es de gran utilidad a la hora de compartir datos entre cliente y servidor.

Para este desarrollo, optamos por usar la herramienta *Visual Studio Code*.

La estructura de la aplicación está basada en al estructura predefinida de proyectos *express*, pero con algunas leves modificaciones, esta estructura será explicada de una manera más exhaustiva en apartados posteriores.

Procederemos a explicar la estructura de la aplicación web, la cual sigue una estructura estándar de proyectos *express* en *node*. Podemos encontrar esta estructura en la Figura 6.2

En primer lugar nos encontramos la carpeta *node_modules* en la que se encuentran todas las extensiones y librerías que hemos usado en el proyecto, esta carpeta se puede generar automáticamente mediante el comando *npm*

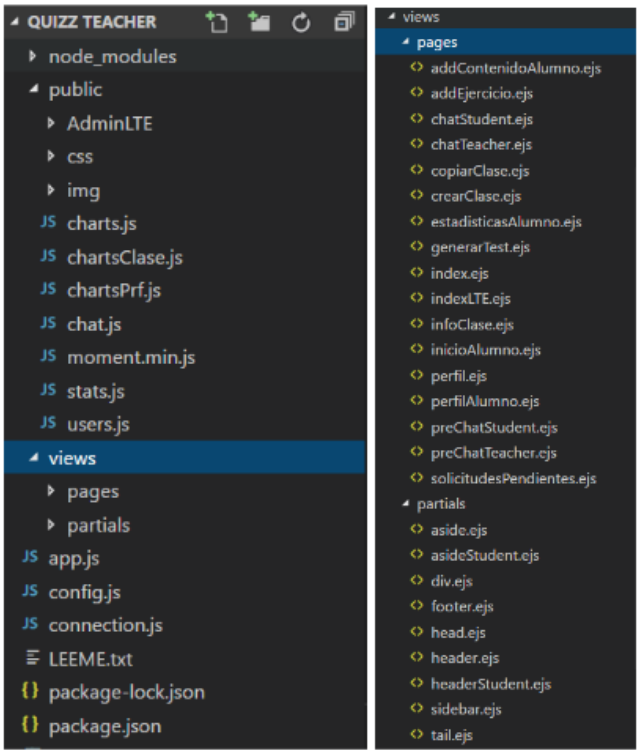


Figura 6.2: Estructura de la aplicación web

install.

A continuación nos encontramos la carpeta public, la cual contiene un conjunto de subcarpetas y archivos formado por:

- La carpeta *AdminLTE* que contiene todos los archivos relacionados con este complemento, el cual es el elemento principal de nuestra interfaz, esta carpeta no deberá ser modificada en ningún momento.
- La carpeta *Css*, la cual contiene las hojas de diseño de nuestra aplicación, que complementan y personalizan la interfaz de la aplicación y del chat respectivamente.
- La carpeta *Img*, en la que almacenamos las imágenes usadas por la aplicación y la carpeta en la que se guardan las imágenes subidas desde la página web mediante el plugin de *multer*.
- Los archivos *charts.js*, *chartsClase.js* y *chartsPrf.js* se encargan de inicializar y personalizar las gráficas de estadísticas y contribuciones, de los alumnos, de las clases, y del perfil de los alumnos visto desde una cuenta tipo profesor respectivamente.

- El archivo *chat.js* contiene toda la lógica de la parte del cliente del chat. El archivo *moment.min.js* es complementario a *chat.js* y se utiliza para su correcto funcionamiento.
- El archivo *stats.js* contiene toda la lógica de las *DataTables* y toda la funcionalidad *JQuery* relacionada con formularios e interacción con el usuario de toda la aplicación web. Este archivo es clave para la subida de solicitudes de ejercicios y contenidos.
- La carpeta *views* en la que se encuentran las vistas de la aplicación, divididas en dos subcarpetas, la carpeta *partials* contiene los elementos comunes de todas las vistas *EJS*, la cabecera, el pie de página, las barras de navegación etc... La carpeta *pages* contiene todas las vistas *ejs* de la aplicación.

Saliendo ya de la carpeta *public*, en el directorio principal de la aplicación, nos encontramos con el archivo *app.js*, el cual es el punto de entrada de la aplicación y su archivo principal donde se importan y unifican todos los complementos que se usarán en la aplicación, se manejan las sesiones, se enrutan las vistas, se implementa la funcionalidad principal y se realizan las llamadas a la base de datos para intercambiar información con las vistas. También es el archivo en el cual se lanza la aplicación web.

Vamos a proceder a explicar más en profundidad este archivo:

En primer lugar se importan los distintos complementos o módulos que se van a utilizar en la aplicación y se configuran las sesiones, usando para esto los datos del archivo *config.js*. También tenemos una serie de configuraciones del modulo express para facilitar el intercambio de información con las vistas o el enrutamiento de éstas.

A continuación tenemos una serie de *middlewares* que se encargarán de que cada usuario solo pueda acceder a las partes de la aplicación que le corresponden, estos *middlewares* impiden, por ejemplo, que un alumno acceda a la información de otro o que un profesor pueda ver datos de un alumno que no está en su clase. Estos *middlewares* se usan como parámetro en las funciones de enrutamiento.

Finalmente, tenemos el cuerpo principal de la aplicación, en la que haciendo uso de *express* se lleva a cabo el enrutado y el intercambio de información entre cliente y servidor, para mostrar su estructura pondremos como ejemplo la página de inicio del profesor, la cual es una operación muy simple que solo muestra datos. Podemos encontrar un diagrama de secuencia de esta operación en la figura 6.3

En esta operación, se entra desde la ruta */perfil*, cuando un usuario introduce esa ruta en el navegador, se comprueba que el usuario de la sesión es de tipo profesor, mediante el *middleware authProf* después se procede a buscar en la base de datos el número de solicitudes pendientes de este, sus clases y sus alumnos.

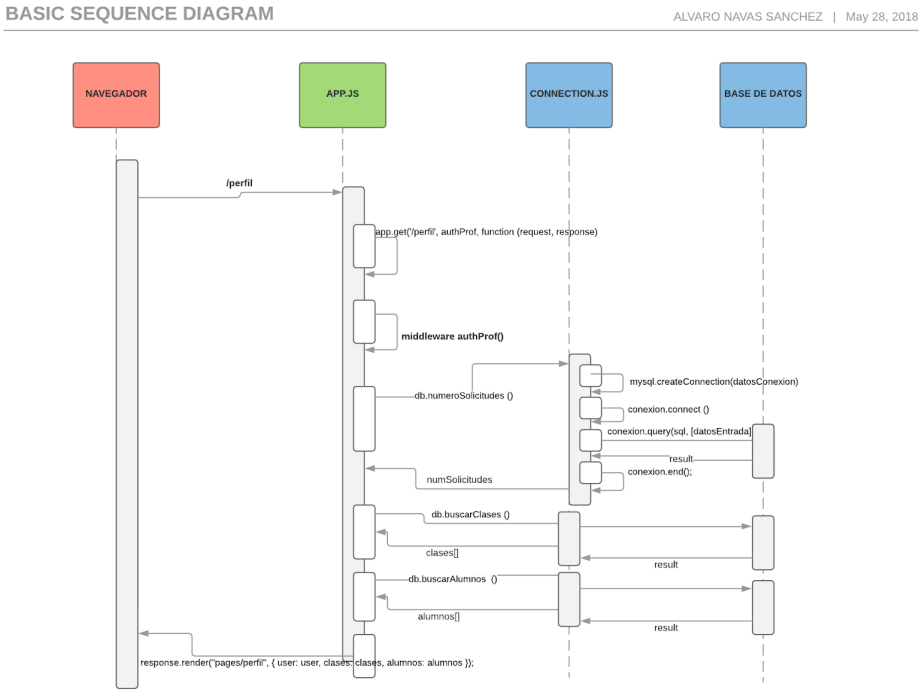


Figura 6.3: Diagrama de secuencia de la operación perfil del profesor

Finalmente, se procede a renderizar el archivo ejs correspondiente, pasándole los datos devueltos por la base de datos, el cual será la vista final que se le mostrará al usuario. No hay que olvidar que a la hora de renderizar y mostrar la vista al usuario se ejecuta el código *Javascript* correspondiente en el navegador usando *JQuery*.

En el caso de operaciones de tipo post, que recojan datos de la vista para insertarlos en la base de datos, la estructura sería muy parecida.

El archivo *config.js* contiene los datos de la conexión y la base de datos, entre estos datos se encuentran la dirección del *host*, el nombre de la base de datos y el usuario y la contraseña para acceder a esta.

En el archivo *connection.js* se realiza la conexión a la base de datos, usando los datos del archivo *config.js* y se interactúa con ella. En este archivo están implementadas todas las funciones de la base de datos y las consultas que realizamos. Desde el archivo *app.js* se llama a una operación del *connection.js*, en esta operación en primer lugar creamos la conexión, usando los datos del archivo *config.js*, después nos conectamos con la base de datos y ejecutamos la consulta *SQL* deseada con los datos de entrada que sean necesarios. Finalmente, cerramos la conexión y, si no hay ningún error, devolvemos el resultado de la consulta.

6.3. Conexiones iniciales, sesiones y acoplamiento a la base de datos

Una vez definida la estructura inicial de nuestra aplicación web, procedimos a realizar la conexión a la base de datos.

Antes de empezar con el desarrollo de las funcionalidades básicas y la interfaz de la aplicación web, empezamos por conectar desde *Node.js* con la base de datos de la aplicación Android, para esto, cargamos en local una copia de la base de datos, y montamos un servidor en *localhost* usando *Xampp* Wikipedia (Xampp). Una vez realizada la conexión, implementamos un formulario de login simple, y un sistema de sesiones, utilizando para ello *Express-session* el cual es un middleware que se encarga de crear y almacenar la sesión de forma local y *Express-mysql-session* el cual nos permite almacenar la sesión actual en base de datos, para así permitir múltiples sesiones de forma simultánea, y poder guardarlas durante el periodo de tiempo seleccionado, antes de que caduquen. Posteriormente procedimos a lanzar la aplicación montada en *express* sobre dicho servidor.

En este punto teníamos una aplicación web capaz de conectarse con la misma base de datos que la aplicación Android, y con un sistema de sesiones compatible con esta, por lo tanto, ya era apta para los usuarios de la aplicación Android. Por lo que procedimos a decidir que herramientas usar para la creación de las vistas.

6.4. Creación de las vistas responsive y su acoplamiento a express

Una de nuestras prioridades respecto a la aplicación web, es que fuese totalmente *responsive*, es decir, que sea completamente compatible con cualquier dispositivo móvil. Por ello, después de barajar varias opciones optamos por usar para las vistas la librería de código abierto *AdminLTE* que ofrece un amplio número de componentes web, los cuales son completamente responsive y reusables.

Aquí surgió nuestro primer problema, *AdminLTE* se basa en vistas *HTML*, y nosotros estábamos usando plantillas *EJS*. *AdminLTE* no está pensado para usarlo con plantillas *EJS*, por lo cual, tuvimos que modificar los ficheros fuente de *AdminLTE* para que fuesen compatibles con nuestra aplicación, cambiándolos de formato y reestructurando el código de tal forma que fuese compatible con el sistema de plantillas.

En este punto teníamos una aplicación funcional y un sistema para crear vistas responsive de forma sencilla, así que decidimos implementar las funcionalidades básicas de la aplicación, usando dicho sistema para construir las vistas según las íbamos necesitando. Para ello, dejamos un esqueleto de

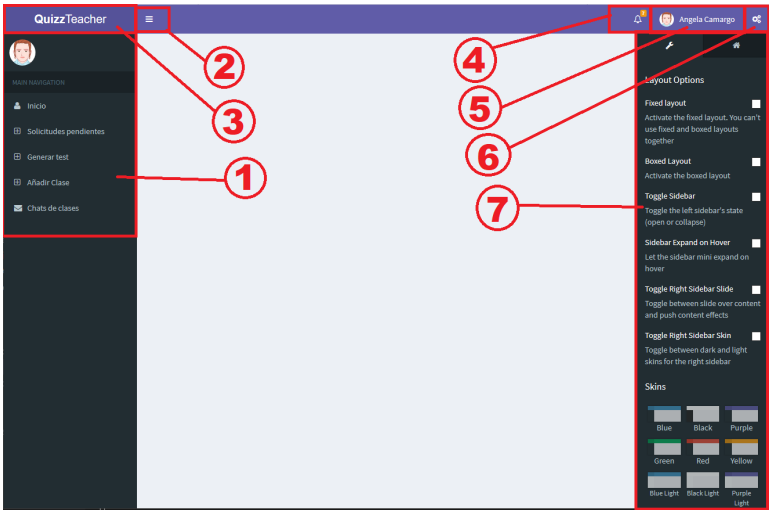


Figura 6.4: Esqueleto de la interfaz de la aplicación web

vistas básico, el cual era común para toda la aplicación. Podemos encontrar este esqueleto en la Figura 6.4 y su versión responsive en la Figura 6.5

La aplicación dispone de un menú de navegación lateral (1), el cual se puede expandir o contraer a gusto pulsando el botón situado a su derecha (2), lo cual es especialmente útil para la navegación desde dispositivos móviles. Desde este menú se accede a las funcionalidades de la web.

También dispone de una cabecera, la cual contiene:

- En la parte izquierda, el logo de la aplicación, el cual al hacer click te redirige a la página principal (3), y un botón que permite expandir o contraer el menú principal (2), del cual ya hemos hablado.

- En la parte derecha dispone de sistema de notificaciones (4), el cual se encarga de decirle al profesor el número de solicitudes pendientes que tienen. Los alumnos no disponen de este botón.

- Continuando hacia la derecha, tiene un botón con el perfil del usuario logueado (5), que al clicarlo se expande, mostrando unos datos simples de este, y el botón de desconexión.

- Por último, dispone de un botón de configuración (6), situado en la parte superior derecha, el cual al clicarlo abre un menú que permite configurar al gusto la interfaz (7), cambiar los colores de ésta y cambiar el posicionamiento de algunos de sus componentes. Lo vemos en la Figura 6.4

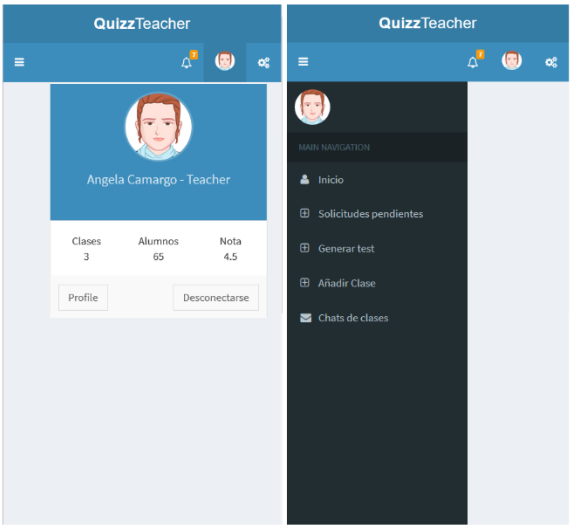


Figura 6.5: Interfaz personalizable

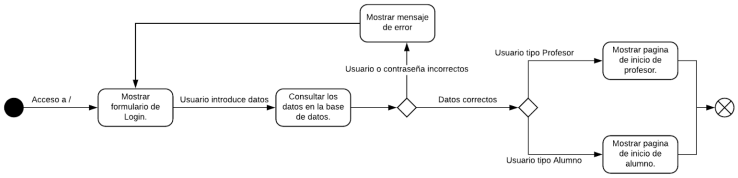


Figura 6.6: Diagrama de actividad del login

6.5. Implementación de las funcionalidades de la aplicación

6.5.1. Sistema de login y sesiones

Lo primero que implementamos fue el sistema de acceso a la aplicación, o sistema de login, el cual comprueba en la base de datos el usuario y la contraseña introducidos, y si son correctos, comprueba el tipo del usuario y dependiendo de si es profesor o alumno, lo redirige a una página con las funcionalidades específicas de su tipo de usuario. Lo podemos ver en la Figura 6.7

Podemos encontrar un diagrama de actividad del inicio de sesión en la Figura 6.6.

Para interactuar con la mayoría de formularios tipo *post*, optamos por usar *body-parser* un middleware que podemos encontrar en el repositorio de

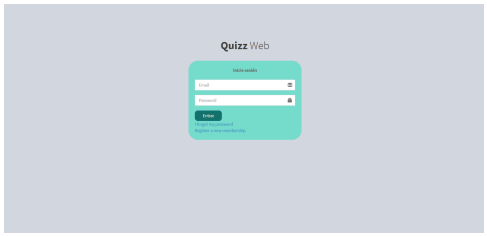


Figura 6.7: Login WebQuizz

npm y que nos permite conseguir datos de los formularios de las vistas de manera muy sencilla.

Módulos de profesor y alumno

Como ya hemos comentado anteriormente, la aplicación se divide en dos partes: la parte para profesores y la parte para alumnos, las cuales tienen una funcionalidad completamente distinta, pero comparten una interfaz común.

A continuación, explicaremos las funcionalidades específicas de cada uno de estos módulos.

6.5.2. Módulo para profesores

6.5.2.1. Inicio

Tras loguearse en la aplicación, se le muestra al profesor dos tablas, una con la información correspondiente a sus clases, y otra con la de sus alumnos.

Para esto, conseguimos la información necesaria de la base de datos mediante llamadas sql. A la hora de mostrar la información optamos usar *Datatables*, un plugin de *JQuery* que permite mostrar información en forma de tablas de una forma flexible. También permite buscar información en dichas tablas de forma inmediata y ordenarlas al gusto, sin necesidad de recargar el navegador, como se puede ver en la Figura 6.10

Podemos encontrar un diagrama de secuencia y uno de actividad de esta operación en las figuras 6.8 y 6.9 respectivamente.

En estas tablas, si hacemos click en el nombre del alumno o de la clase, accedemos a una vista con sus datos específicos, de forma extendida.

En la vista específica de las clases, se muestra la información básica de la clase, como su nombre, código, curso, profesor, la nota media de los test realizados en esta clase desde la aplicación Android, número de alumnos, y número de ejercicios y contribuciones realizadas por estos. También se nos muestra las secciones que tiene la clase.

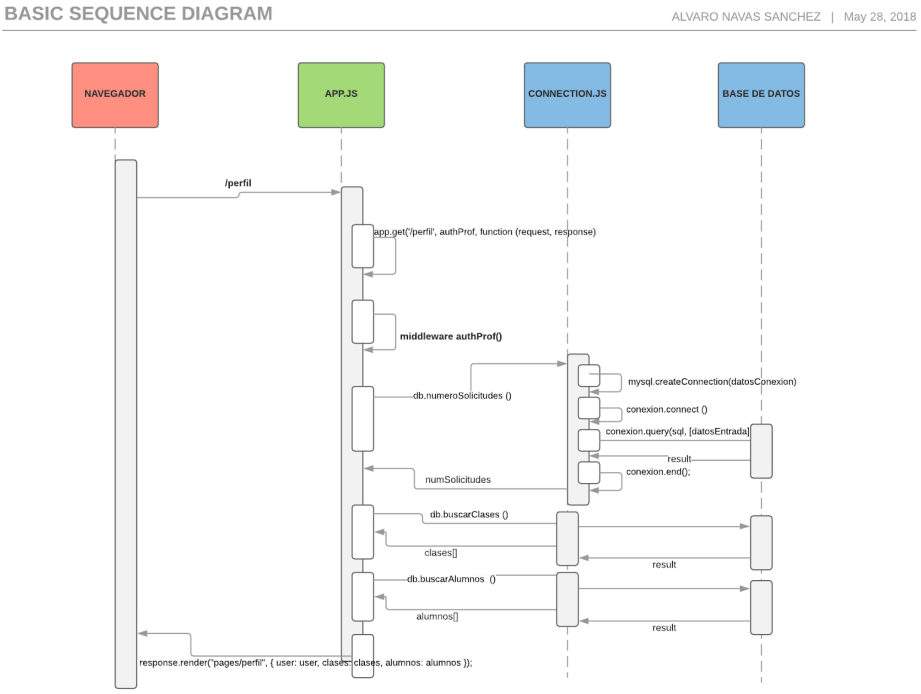


Figura 6.8: Diagrama de secuencia de la operación perfil del profesor

A parte de esto, se muestran una serie de *tabs* (Milán, 2010) los cuales al hacer click sobre ellos, nos muestran más funcionalidades.

A parte de esto, se muestran una serie de tabs, los cuales al clicar sobre ellos, nos muestran más funcionalidades:

Estadísticas y contribuciones de la clase: se muestran una serie de estadísticas simples, recogiendo los datos de los test realizados y las aportaciones hechas por los alumnos de la clase y utilizando la librería *Charts.js*, la cual nos permite convertir estos datos en gráficas responsive como las mostradas en la Figura 6.11. En concreto mostramos una gráfica de barras con las notas medias de la clase en cada contenido, cuya nota se saca de los test realizados por los alumnos en la aplicación Android y una gráfica circular, que muestra las cantidades de aportaciones aceptadas o rechazadas para contenidos y ejercicios. Estas aportaciones, son realizadas por medio de la aplicación web.

Queremos destacar la complejidad de esta clase de operaciones a nivel de base de datos, ya que al tener una estructura tan amplia y una cantidad tan grande de tablas, las consultas a realizar son bastante complejas, por ejemplo, la consulta para obtener las estadísticas de una determinada clase es la siguiente:

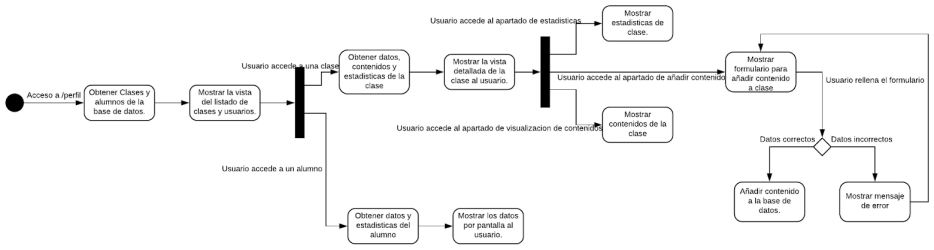


Figura 6.9: Diagrama de actividad de inicio del profesor

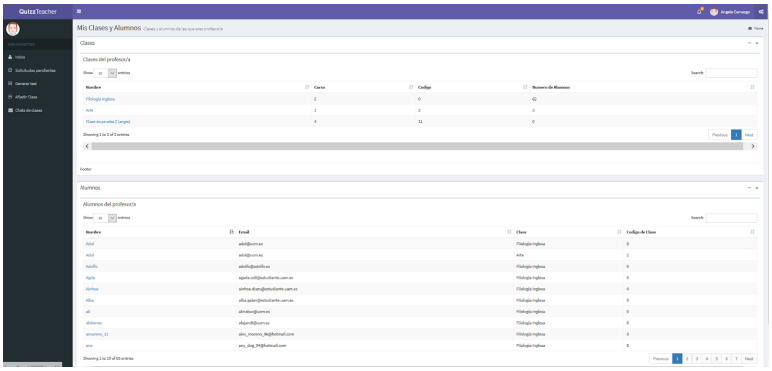


Figura 6.10: Inicio para el profesor

```
1 SELECT aq.questionId, q.statement , q.type, COUNT(CASE WHEN aq.
    correct=1 THEN 1 END) AS numAciertos, COUNT(CASE WHEN aq.
    correct=0 THEN 1 END) AS numFallos , c.name as contentName
    , s.name as sectionName FROM answer_question aq, question q
    , test t, content_section cs,content c, section s WHERE aq.
    questionId = q.id and q.test_id = t.id and t.content_id= cs
    .id_content and cs.id_section = s.id and s.class_id = ? and
     c.id =  t.content_id group by q.id
```

Aun así, es importante destacar que la estructura de la base de datos y de la interfaz se ha diseñado de manera que permita añadir todo tipo de estadísticas complejas de manera sencilla.

Contenidos de la clase: Se muestran los contenidos de la clase y la sección a la que pertenecen usando la librería *Datatables*. Además, al hacer click en el campo "Descripción" se despliega un pop-up con la descripción del contenido. Cabe destacar que al buscar en la tabla, la búsqueda se realiza también en la descripción del contenido tal como se puede ver en 6.12

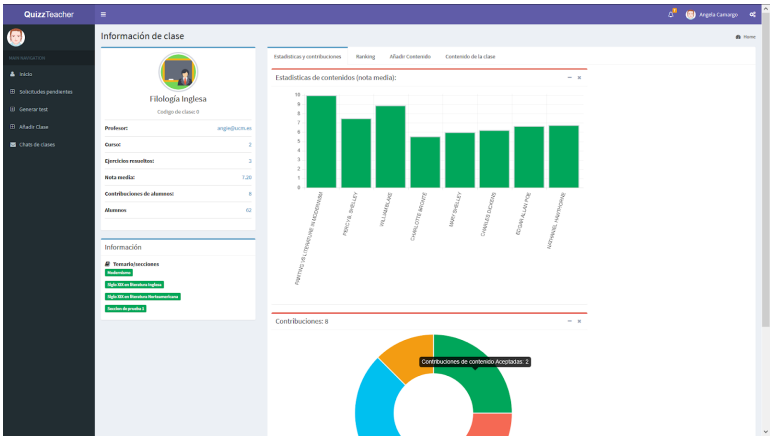


Figura 6.11: Estadísticas de clase

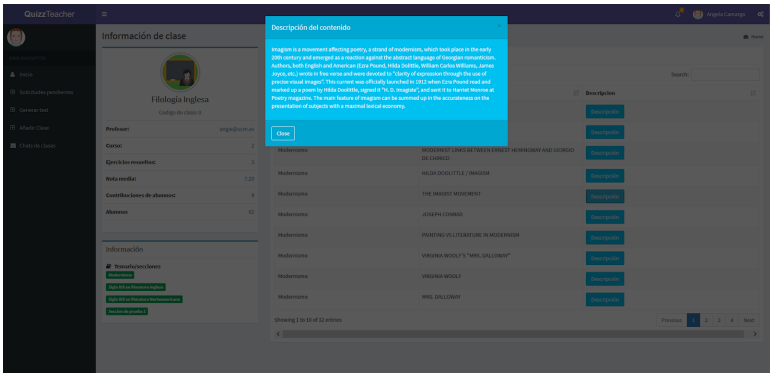


Figura 6.12: Descripción de contenido de clase

Añadir Contenidos a la clase: Se muestran dos formularios que permiten o bien añadir nuevos contenidos a una sección, o añadir nuevas secciones a la clase. Estos contenidos y secciones serán completamente accesibles desde la aplicación Android en el momento en que se añadan. Lo podemos ver en 6.13

En la vista específica del alumno, se nos muestra la información de este, su nombre, correo electrónico, las clases en las que está registrado, y el número de contribuciones y nota media de este. La nota media se obtiene de los test realizados por el alumno desde la aplicación Android. Podemos encontrar un ejemplo de esto en la Figura 6.14

Al igual que para las clases, se muestran las estadísticas de este alumno, en forma de gráficas, una gráfica de barras que muestra la nota media del alumno en los contenidos en los que haya realizado tests, y una gráfica circular, que muestra las cantidades de aportaciones aceptadas o rechazadas para

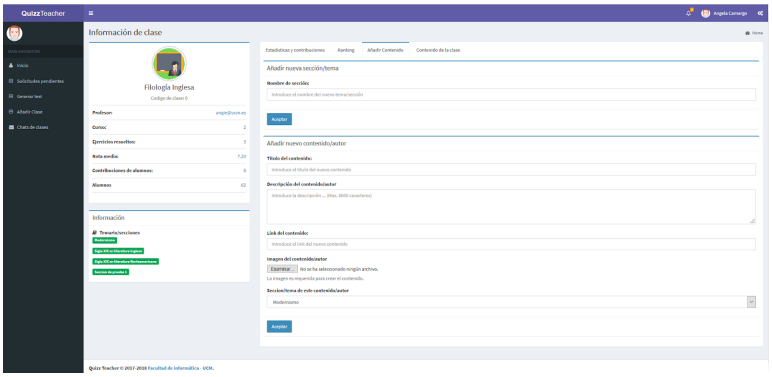


Figura 6.13: Añadir contenido a una clase desde una cuenta tipo profesor

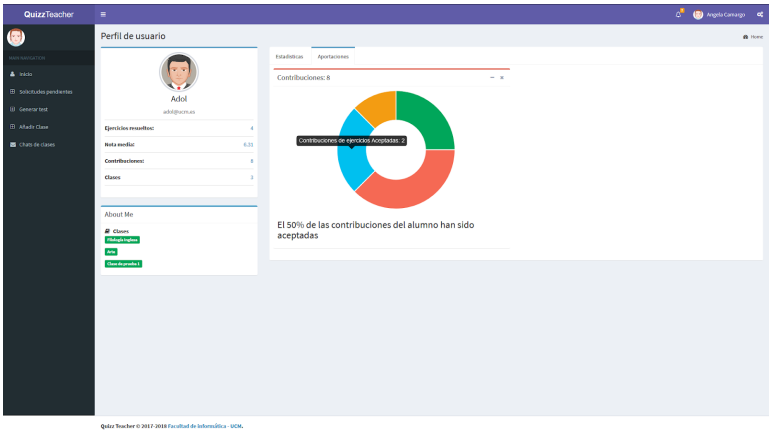


Figura 6.14: Vista específica de alumno desde una cuenta tipo profesor

contenidos y ejercicios por el alumno. Estas aportaciones, son realizadas por medio de la aplicación web.

6.5.2.2. Añadir clase

Se le muestra al profesor un formulario para crear una nueva clase como el mostrado en la Figura 6.15. Una vez creada la clase le devuelve al profesor un código, el cual usarán los alumnos, a través de la aplicación Android, para unirse a dicha clase y tener acceso a sus contenidos y ejercicios. La operación la podemos ver en el diagrama de secuencia de la Figura 6.16

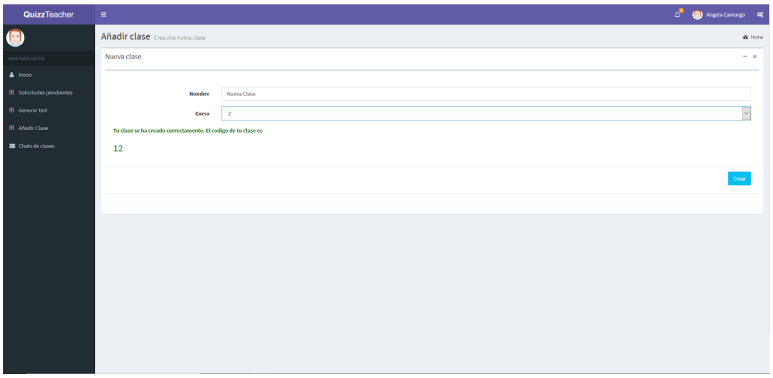


Figura 6.15: Formulario Añadir clase

6.5.2.3. Solicitudes pendientes

En este apartado se le muestra al profesor una serie de solicitudes, previamente realizadas por alumnos de su clase mediante la aplicación web, divididas en dos tablas, una para ejercicios y otra para contenidos.

Las solicitudes contendrán un ejercicio o un contenido, los datos del alumno que realiza la solicitud y en caso de ser una solicitud de ejercicio, el contenido al que pertenece dicho ejercicio, para posteriormente asociarlo a un test dentro de dicho contenido. Podemos encontrar un ejemplo de esto en la Figura 6.17

El profesor podrá revisar dichas solicitudes, clicando en el campo "Descripción" el cual se desplegará en forma de *pop-up* permitiendo revisar el ejercicio o contenido enviado por el alumno, como podemos ver en la Figura 6.18. Cuando un profesor acepta o rechaza una solicitud se guardan los datos del alumno y del contenido o ejercicio en la base de datos para posteriormente generar estadísticas.

En caso de aceptar las solicitudes de contenidos, estos se añadirán a la clase, y serán completamente accesibles por parte de los alumnos de dicha clase desde la aplicación Android en el mismo momento de ser aceptados.

En caso de aceptar una solicitud de ejercicio, el ejercicio se incluirá en la base de datos, pero no formará parte de ningún test por el momento, por lo que no estará disponible en la aplicación Android hasta que se añada como parte de un test, lo cual será posible desde otra vista de la aplicación web. En el caso de rechazar una solicitud de contenido o ejercicio esta solicitud será completamente eliminada de la base de datos.

Podemos encontrar un diagrama de actividad de esta operación en la Figura 6.19, en el que se describen tanto la participación del alumno como la del profesor en esta operación.

De la misma manera podemos encontrar un diagrama de secuencia de

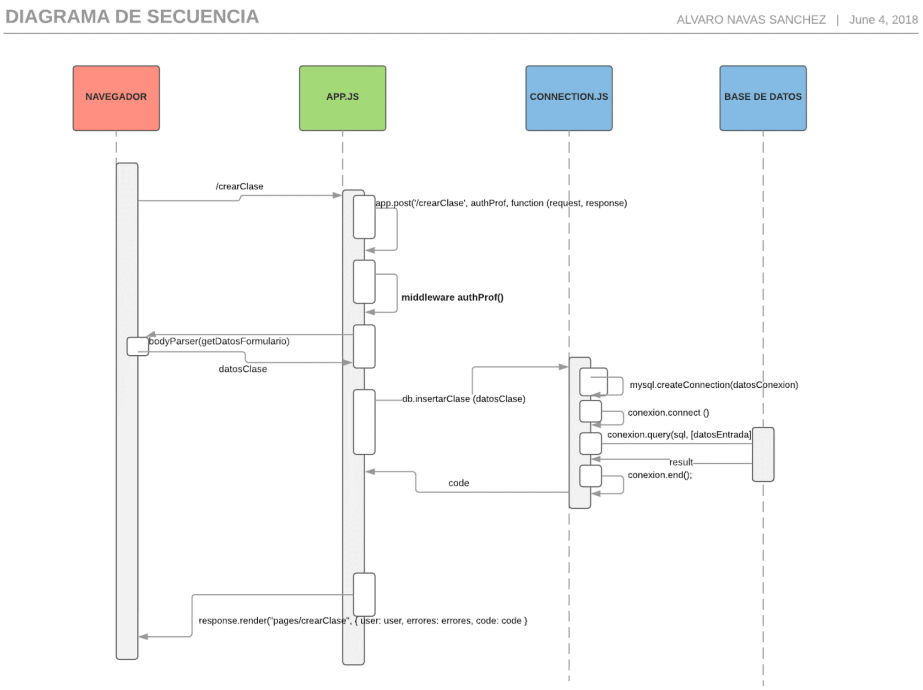


Figura 6.16: Diagrama de secuencia de crear clase

la operación aceptar solicitud de contenido en la Figura 6.20, el resto de funciones de aceptar o rechazar solicitudes son equivalentes.

6.5.2.4. Generar test

En este apartado se le muestran al usuario una lista de los contenidos que tienen asociados ejercicios sin test, es decir, ejercicios enviados por alumnos que ya hayan sido aceptados, pero que todavía no han sido incluidos en ningún test.

Cuando el usuario selecciona alguno de esos contenidos, se le muestra cuántos ejercicios sin test hay en dicho contenido, y, solo si, hay como mínimo 5 ejercicios, se le permite generar un test, que podrá tener un número a elegir por el usuario entre 5 y 10 ejercicios. Se ha elegido esta extensión para que los test queden de un tamaño razonable y su uso sea más cómodo y variado desde la aplicación Android. En la Figura 6.21 y en la Figura 6.22 podemos ver ambos ejemplos, uno para un contenido con ejercicios insuficientes y otro para uno con ejercicios suficientes.

Una vez seleccionado el número de ejercicios y el contenido sobre el que se quiere realizar el test, el usuario podrá generar un test en el que se incluirán de manera aleatoria el número de ejercicios seleccionados. Los test se forman

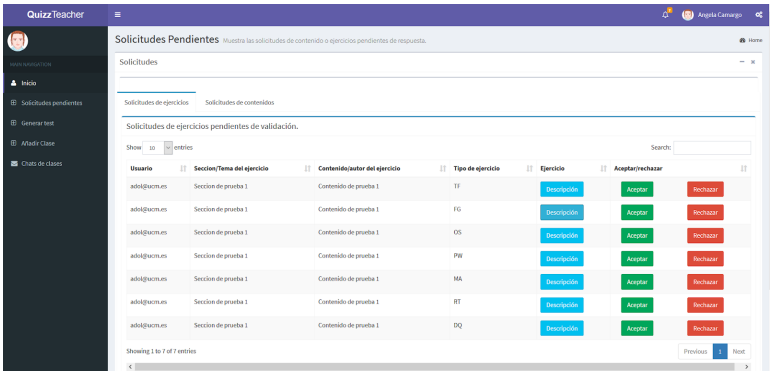


Figura 6.17: Solicitudes de ejercicio pendientes

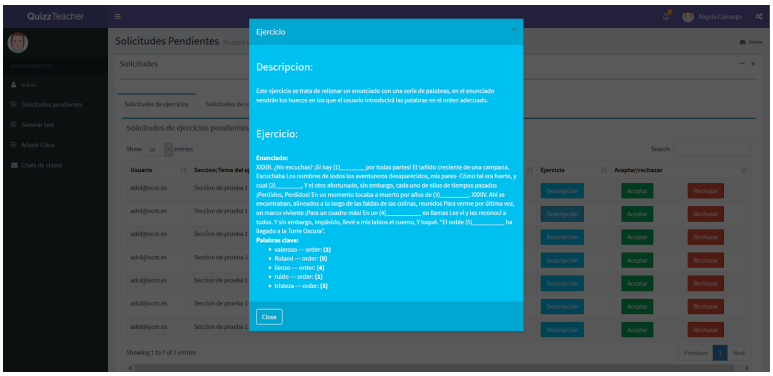


Figura 6.18: Descripción de una solicitud de ejercicio

de manera aleatoria para facilitar el proceso y hacerlo lo más cómodo posible al profesor, de esta manera también conseguimos test variados.

Una vez generado el test, y asignado a un contenido, se guardará en la base de datos y este será totalmente visible en la aplicación web para todos los alumnos de su clase. Los ejercicios que no sean elegidos se conservarán intactos y sin estar asignados a ningún test para su posterior incorporación.

6.5.3. Módulo para alumnos

El módulo para alumnos está pensada principalmente para que estos puedan añadir contenidos a la aplicación, creándolos a través de formularios los cuales el profesor recibirá en forma de solicitud, que podrá aceptar o rechazar. A continuación explicaremos las distintas funcionalidades de este módulo.

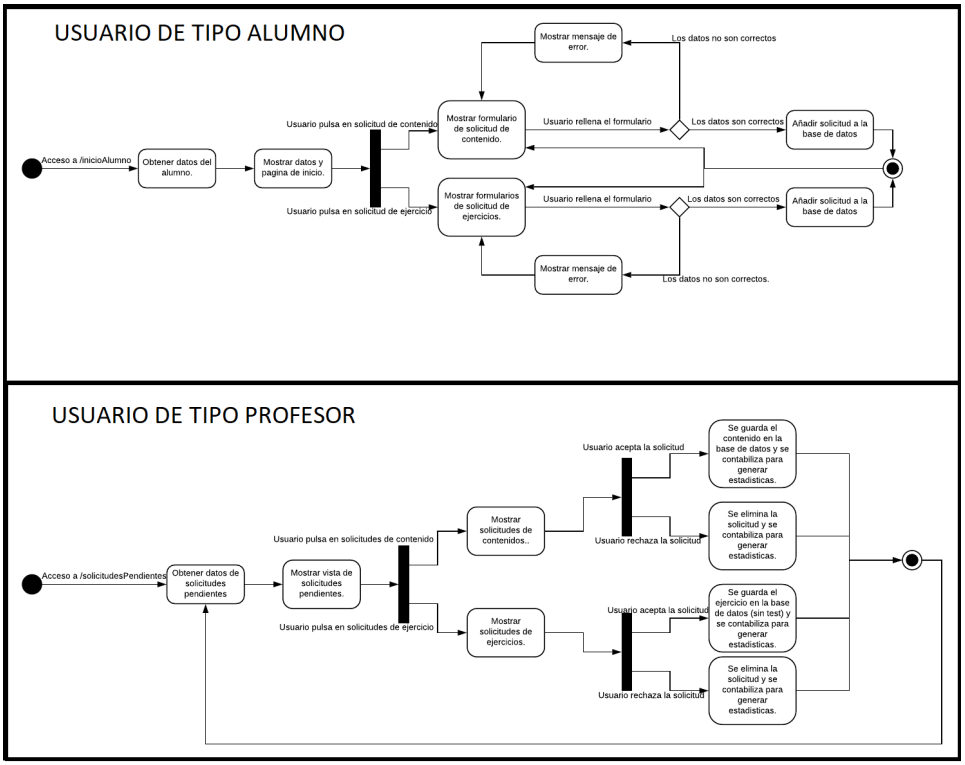


Figura 6.19: Diagrama de actividad de solicitudes

6.5.3.1. Inicio/añadir ejercicios o contenido

En este apartado se muestra la información del alumno, su nombre, mail, las clases en las que está registrado y el número de contribuciones y nota media de éste.

También se le muestra un formulario con las secciones a las que tiene acceso y las clases a las que pertenece. El alumno deberá elegir si desea añadir un nuevo contenido o un nuevo ejercicio y la sección y clase en la que desea hacerlo. Podemos encontrar este formulario en la Figura 6.23

Dependiendo de lo que elija será redirigido a una vista que contenga los formularios pertinentes a su elección.

En el caso de querer añadir un nuevo contenido al alumno se le muestra un formulario simple, para añadir un contenido a la sección seleccionada. Una vez rellenos los datos y enviado el formulario, este se guardará en la base de datos en forma de solicitud de contenido y quedará a la espera de que el profesor de la clase lo acepte o rechace, como se muestra en la Figura 6.24

En el caso de querer añadir nuevos ejercicios, en forma de solicitud, se le muestran una serie de formularios, uno para cada tipo de ejercicio existente.

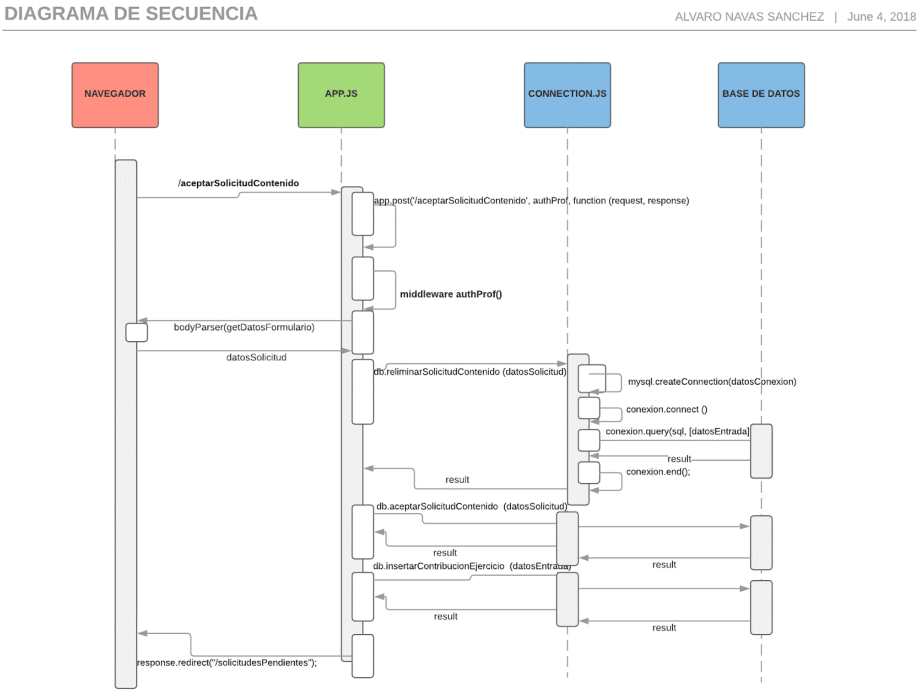


Figura 6.20: Diagrama de secuencia de la operación aceptar solicitud

Para cada formulario, se le mostrará al usuario una descripción del tipo de ejercicio y un ejemplo, para facilitar el uso de estos formularios y de los campos a rellenar.

Estos formularios son mucho más complejos que el resto de formularios de la aplicación. Para empezar, tuvimos que crear nuevas tablas en la base de datos para guardar las solicitudes, una por cada tipo de ejercicio.

Algunos de los campos de los formularios son variables, es decir, que pueden tener un número a seleccionar de respuestas, o de opciones a elegir. Para esta clase de formularios usamos *JQuery*para modificar el formulario en función de las necesidades del usuario, sin necesidad de recargar la página. También hacemos uso de las plantillas *EJS* para mostrar datos sacados de la base de datos de manera dinámica y acoplarlos como parte del formulario, y comunicando las plantillas *EJS* con *JQuery* conseguimos hacer formularios capaces de guardar estructuras de datos complejas en la base de datos. Estos formularios son muy distintos entre sí, ya que cada tipo de ejercicio contiene una estructura única.

En los formularios también se dará a elegir entre los contenidos que pertenezcan a la sección seleccionada, ya que cada ejercicio debe ir relacionado con un contenido para, posteriormente, incorporarse a un test relacionado

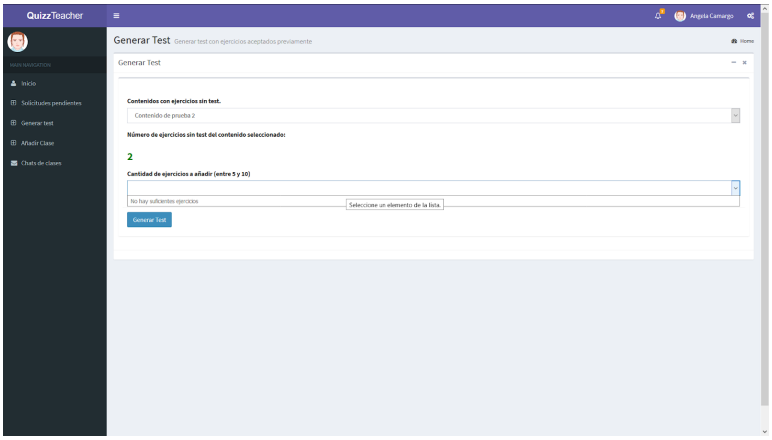


Figura 6.21: Generar test con contenidos insuficientes

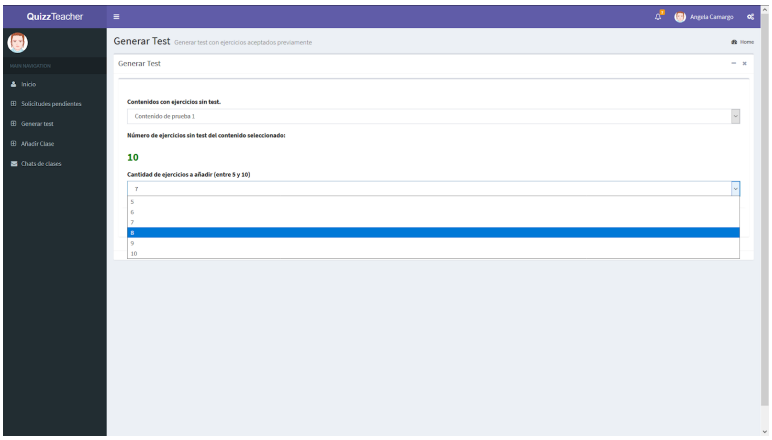


Figura 6.22: Generar test con contenidos suficientes

con dicho contenido, y de esta manera adaptarse a la estructura de la aplicación Android. Podemos ver un ejemplo de algunos de estos formularios en la Figura 6.25 y en la Figura 6.26

Podemos encontrar un diagrama de actividad de esta operación en la Figura 6.19, en el que se describen tanto la participación del alumno como la del profesor en esta operación.

6.5.3.2. Estadísticas

Al igual que en la vista de perfil de alumno del módulo del profesor, se nos muestra la información de este, y también se muestran las estadísticas de este alumno, en forma de gráficas, una gráfica de barras que muestra la

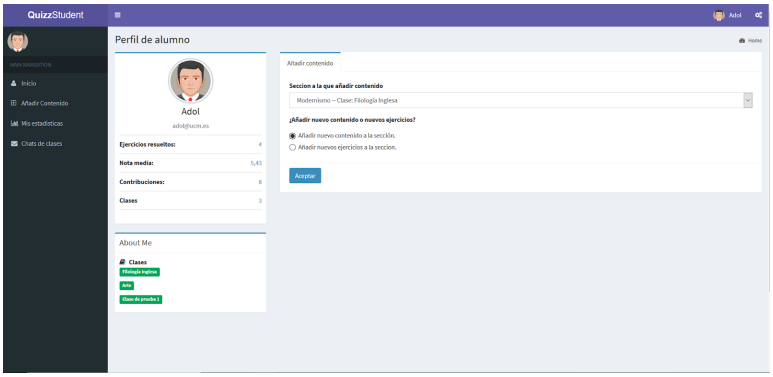


Figura 6.23: Vista de Inicial de un usuario tipo alumno

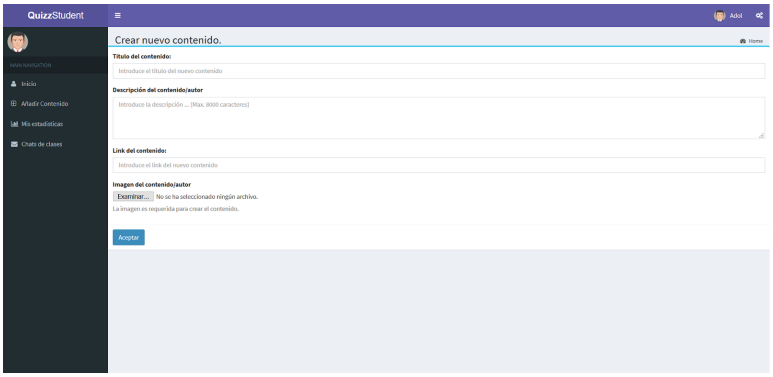


Figura 6.24: Formulario de solicitud de contenido

nota media del alumno en los contenidos en los que haya realizado tests, y una gráfica circular, que muestra las cantidades de aportaciones aceptadas o rechazadas para contenidos y ejercicios por el alumno. Estas aportaciones, son realizadas por medio de la aplicación web.

6.5.4. Chat

Esta funcionalidad es común para profesores y alumnos. Es un chat simple, interno de la clase, al cual solo podrán acceder los alumnos y el profesor de dicha clase. Se puede utilizar como un canal de comunicación con fines tanto didácticos como lúdicos por el cual esperamos fomentar el uso de la aplicación web. Cada clase dispone de su propia sala de chat.

Para desarrollar el chat, optamos por usar *Socket.io*, que nos permite manejar eventos en tiempo real mediante una conexión *TCP* y todo ello en *JavaScript*.

La parte más complicada fue crear un sistema de chats, el cual tuvie-

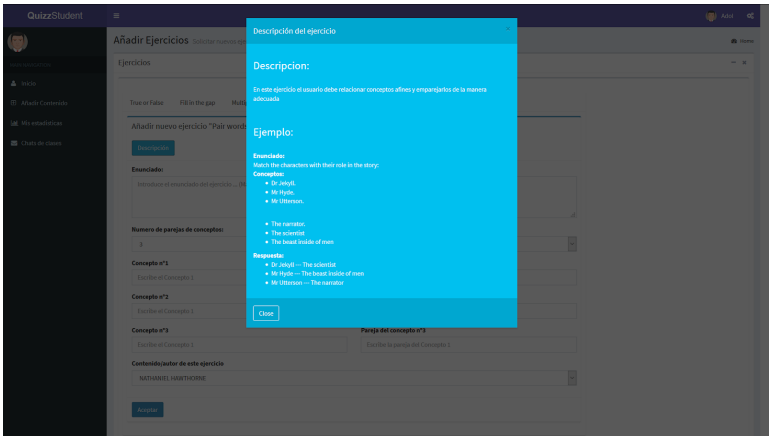


Figura 6.25: Solicitud de ejercicio Pair words con campo de ejemplo abierto y 3 conceptos

se varias salas, y que cada una de estas salas solo fuera accesible para un grupo de usuarios específicos, los cuales son los alumnos de dicha clase y su profesor. También aportamos al chat una interfaz amigable y *responsive*, que permitiera distinguir fácilmente los mensajes enviados de los recibidos, y asociarlos a su emisor. Podemos ver un ejemplo de este chat en la Figura 6.27

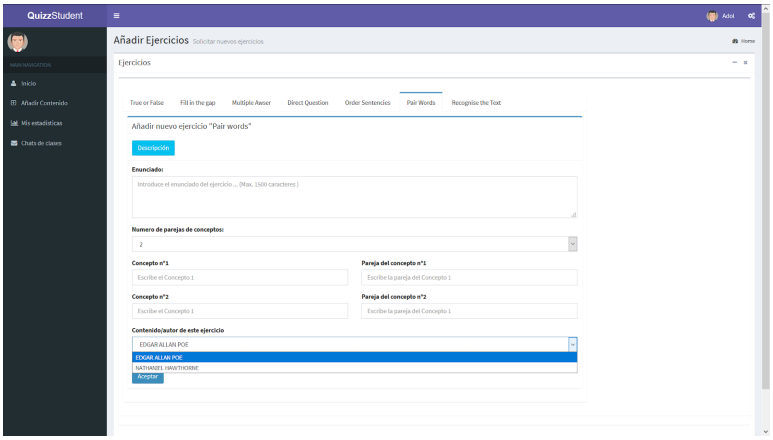


Figura 6.26: Solicitud de ejercicio Pair words con campo de ejemplo cerrado y 2 conceptos

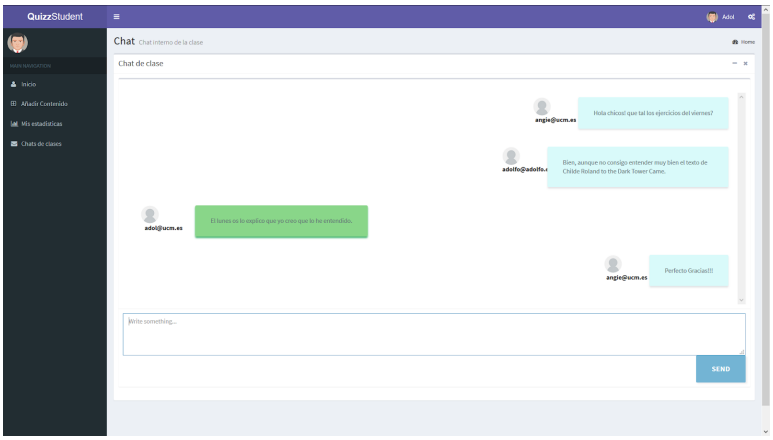


Figura 6.27: Chat de la Aplicación web

Capítulo 7

Parseador de documentos Word

El único sistema seguro es aquel que está apagado en el interior de un bloque de hormigón protegido en una habitación sellada rodeada por guardias armados

Gene Spafford, experto en seguridad

7.1. Introducción

Cuando iniciamos este trabajo, dado que partíamos de un trabajo ya desarrollado, estuvimos un tiempo estudiando las distintas posibilidades de complementar el trabajo ya desarrollado. Como se ha explicado anteriormente, uno de los puntos flacos de la aplicación *Quizz* era la ausencia de herramientas que permitiesen añadir, editar o eliminar contenido de la aplicación.

Debido a esos problemas, la forma que tenían los usuarios de la aplicación de editar contenidos era rellenar un documento Word y enviárselo a los desarrolladores, que realizarían las correspondientes *queries* sobre la base de datos para insertar, modificar o eliminar contenido; así como crear nuevos test para poder utilizarlos en la aplicación.

Para los usuarios de la aplicación, de un perfil tecnológico bajo de media, les resultaba muy cómoda esta manera de gestionar los contenidos. El problema radicaba en el hecho de que siempre se requería de la participación de los desarrolladores de la aplicación; los usuarios de la misma no podían gestionar los contenidos de la misma, ni los alumnos ni el profesor.

Por estas razones, pensamos que era interesante desarrollar un parseador de documentos Word, para que los usuarios de la aplicación pudiesen gestionar el contenido de la aplicación sin necesitar a los desarrolladores.

7.2. Pasos previos

Antes de acometer el desarrollo de la parte del parseador realizamos un estudio exhaustivo de la mejor manera de llevar a cabo la implementación. Entre otras, barajamos las siguientes opciones:

- Realizar la implementación en Java. Esto se haría mediante el uso de la biblioteca de Apache (Apache, 2018) POI (Apache, 2010)
- Realizar la implementación en Python, para lo que se podría utilizar la biblioteca *python-docx* (Canny, 2013)
- Realizar la implementación en Javascript. Esto podía hacerse mediante las bibliotecas *docx-extractor*, *docx4js* o *word-extractor*; todas ellas del entorno de ejecución *Node.js*

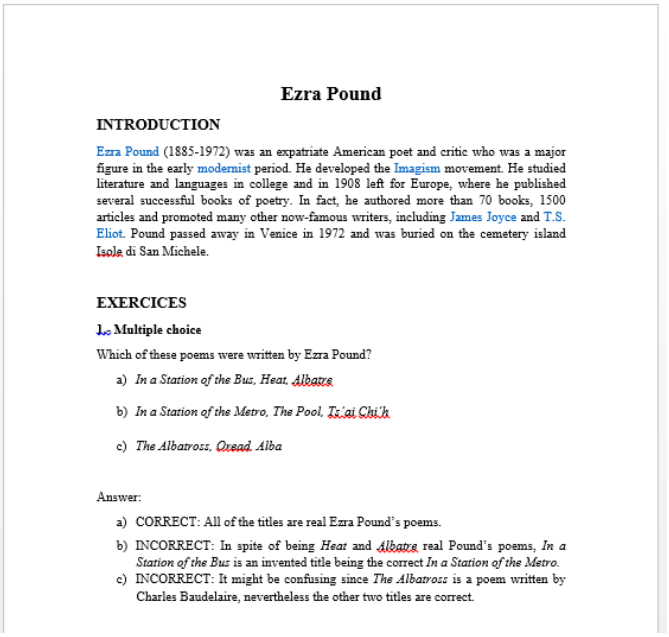
Al final nos decantamos por utilizar *Javascript* para el desarrollo por tener varias bibliotecas para la implementación y por creer que sería más sencilla su integración con el portal web, al estar también desarrollado en *Javascript*.

7.3. Estructura de los documentos

Antes de comenzar a implementar el parseador, hicimos el estudio de una serie de documentos de ejemplo, facilitados por nuestro director de TFG, de cara a analizar la mejor forma de abordar el desarrollo. De esa forma pudimos ver las distintas secciones que integraban estos documentos. Viendo varios de ellos, pudimos ver que su estructura es la siguiente:

1. En primer lugar el nombre del autor o tema sobre el cual va a tratar el documento.
2. Acto seguido, el literal "INTRODUCTION" -ciertos patrones de texto o palabras que indican la sección que figurará a continuación son muy importantes en la implementación del parseador- seguido de uno o varios párrafos que narrarán una introducción sobre el tema tratado, y que será importante de cara a contestar correctamente los ejercicios propuestos.
3. Tras la introducción, llegará la parte de los ejercicios. Ésta comenzará con el literal "EXERCICES", tras lo cual comenzarán los ejercicios propuestos sobre el tema, con un formato que se explicará detenidamente más adelante.

En la figura 7.1 podemos ver el comienzo de uno de estos documentos, incluyendo uno de los tipos de ejercicios que vienen incluidos en la aplicación *Quizz*.



Ezra Pound

INTRODUCTION

Ezra Pound (1885-1972) was an expatriate American poet and critic who was a major figure in the early modernist period. He developed the Imagism movement. He studied literature and languages in college and in 1908 left for Europe, where he published several successful books of poetry. In fact, he authored more than 70 books, 1500 articles and promoted many other now-famous writers, including James Joyce and T.S. Eliot. Pound passed away in Venice in 1972 and was buried on the cemetery island Isola di San Michele.

EXERCICES

1. Multiple choice

Which of these poems were written by Ezra Pound?

a) *In a Station of the Bus, Heat, Albatre*

b) *In a Station of the Metro, The Pool, Ts'ai Chi'h*

c) *The Albatross, Oread, Alba*

Answer:

a) CORRECT: All of the titles are real Ezra Pound's poems.

b) INCORRECT: In spite of being *Heat* and *Albatre* real Pound's poems, *In a Station of the Bus* is an invented title being the correct *In a Station of the Metro.*

c) INCORRECT: It might be confusing since *The Albatross* is a poem written by Charles Baudelaire, nevertheless the other two titles are correct.

Figura 7.1: Documento Word, con introducción y ejercicio de tipo "Multiple Answer"

7.4. Problemas de implementación

Tras todo el análisis previo, decidimos probar la biblioteca *docx4js*, mencionada anteriormente. Esto fue debido a que leímos que tenía grandes posibilidades como la de, en un futuro, parsear más tipos de archivos ofimáticos lalalic GitHub (2017). Finalmente tuvimos que desechar esta opción porque analizando su uso y haciendo pequeñas pruebas con ello, nos pareció demasiado complejo de utilizar y creímos que a la larga nos resultaría difícil el desarrollo.

Después hicimos un segundo intento con la biblioteca *docx-extractor*, que parecía más sencillo y con bastantes posibilidades. Lamentablemente descubrimos que este paquete era perfecto para extraer ciertas características del documento, pero no para extraer el contenido del documento en sí. La sencillez se puede ver en el siguiente fragmento de código, que permite extraer el autor del documento:

```
1 var dxe = require('docx-extractor');
2 dxe.getAuthor('doc', function(data){
3     console.log(data);
4 });
```

En la figura 7.2 se puede ver el resultado de la ejecución del código anterior en la consola de *NetBeans*.

Figura 7.2: Autor del documento 'doc'

Finalmente, y dado que las demás opciones contempladas no fueron finalmente viables, decidimos utilizar *word-extractor*. Esta biblioteca tiene como punto positivo que mantiene exactamente los mismos caracteres separadores que el documento original -otras bibliotecas a las que no referenciamos en este trabajo, por su poca utilidad para lo que necesitábamos, realizaban el parseo interpretando todos los caracteres separadores (saltos de línea, tabuladores, espacios en blanco...) como un espacio en blanco; lo que hacía imposible su uso para este cometido-. Su principal punto negativo es que no es compatible con documentos de extensión *docx* y sólo es capaz de parsear documentos en formato *doc*.

7.5. Formato específico de parseo de los ejercicios

Como se mencionaba en capítulos anteriores, hubo que hacer un estudio previo de la aplicación Android de cara a saber qué aspectos desarrollar e implementar que ayudasen a la mejora de la misma. Entre las cosas que se estudiaron con detalle se encuentra el modelo de datos. Esto era muy importante de cara a saber qué era necesario extraer de cada documento Word.

El parseador está basado en su mayor parte en un sistema de reconocimiento de patrones concretos de texto. Hay determinadas cadenas de texto que deben ser literales -como los ejemplos anteriormente descritos de "INTRODUCTION" y "EXERCICES"- porque para encontrar los elementos de ese ejercicio, hay que buscar previamente este patrón. Estas cadenas literales, además de lo mencionado antes, son aquellas que se refieren a los nombres de los ejercicios, y a las respuestas de los mismos; en caso de que éstas sean simples como "CORRECT", "INCORRECT", "TRUE" o "FALSE".

Su estructura, a nivel de funcionamiento puede verse en el diagrama de actividad de la figura 7.3. Se trata de una sentencia condicional que, en su interior, comprueba si existen todos los distintos tipos de ejercicios en el documento. Esto lo hace comprobando la cadena exacta con el nombre del ejercicio. En caso de no encontrarlo -un documento no tienen por qué contener todos los tipos de ejercicios-, pasa al siguiente, y así hasta llegar al final del documento.

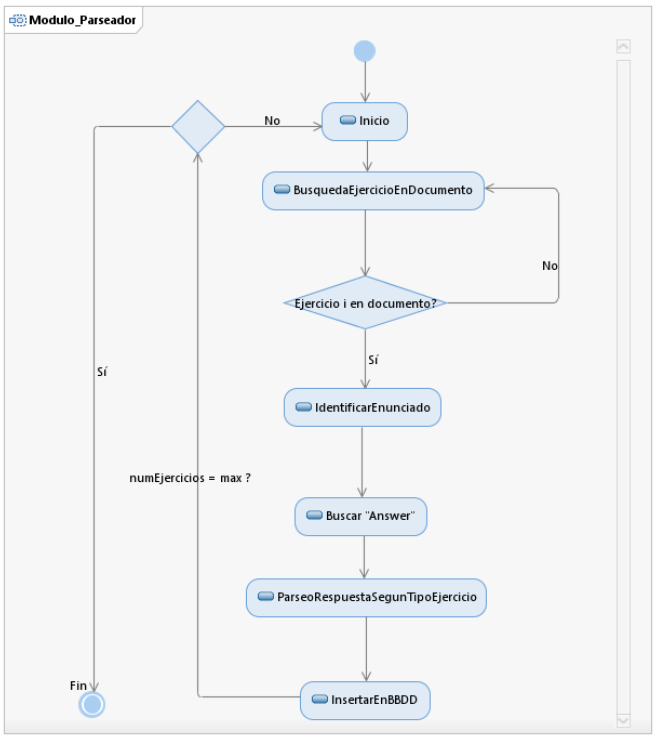


Figura 7.3: Diagrama de actividad del funcionamiento del Parseador de documentos

7.5.1. Parseo de ejercicios

A continuación explicaremos con más detalle cómo se ha realizado el parseo de cada tipo de ejercicio.

7.5.1.1. Multiple answer

En *Multiple answer* tenemos que elegir, de un conjunto de afirmaciones sobre el tema en cuestión, las que se consideren correctas. Para llevar a cabo el parseo de este ejercicio era importante situar el cursor en el caracter siguiente al caracter salto de línea posterior al nombre del ejercicio (algo que generalmente se repetirá en todos los ejercicios) para, llegando hasta el final de la línea, extraer el enunciado del ejercicio.

Bucamos la cadena "Multiple answer" y guardamos su posición en una variable *pos*. Si dicha variable *pos* devuelve un valor entero diferente de -1, significa que tenemos un ejercicio de tipo "Multiple answer". Una vez lo sabemos, sacamos la longitud de la cadena y eliminamos la parte del texto total *body* ya tratada, para después situarnos en el caracter siguiente

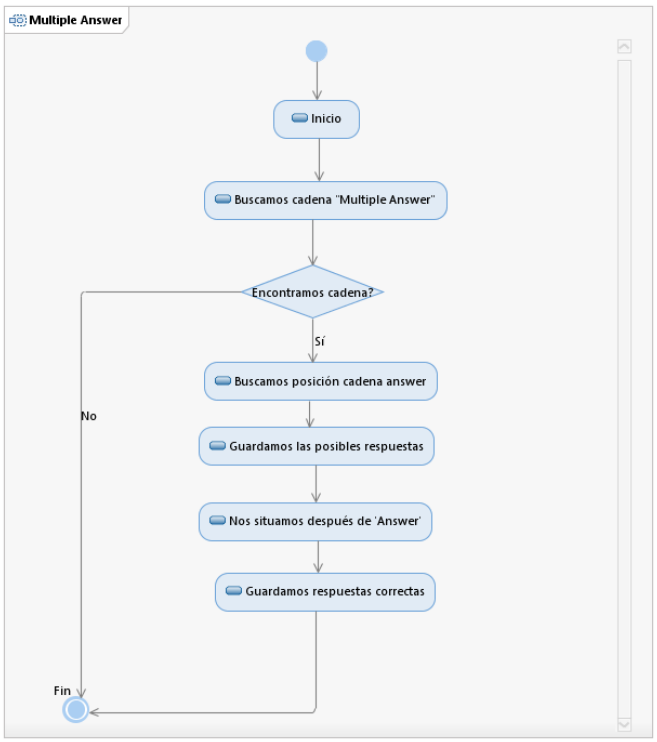


Figura 7.4: Diagrama de actividad ejercicio "Multiple answer"

al del salto de línea posterior al nombre del ejercicio. Una vez ahí, sabemos que desde ese primer caracter de la siguiente línea hasta el final tenemos el enunciado de la pregunta.

A partir de aquí, este tipo de ejercicio se divide en dos partes, las preguntas y las respuestas con su consiguiente explicación. Para ello, una vez situados en el comienzo de la primera pregunta, buscaremos la cadena "Answer:". Así sabremos el fin de las preguntas y el comienzo de las respuestas. En el caso de las preguntas, bastará con ir buscando el carácter final de línea antes de llegar a dicha cadena "Answer". En cuanto a las respuestas, será un poco diferente, teniendo que buscar primero las cadenas "INCORRECT" y "CORRECT" que indiquen si la pregunta es o no correcta, para después extraer también las explicaciones a las respuestas. El procedimiento para identificar estas respuestas es el mismo que para las demás cadenas. En la Figura 7.4 lo podemos ver.

7.5.1.2. True or false

El procedimiento para identificar las opciones en este tipo de preguntas es prácticamente el mismo al empleado en *Multiple answer*. Después de en-

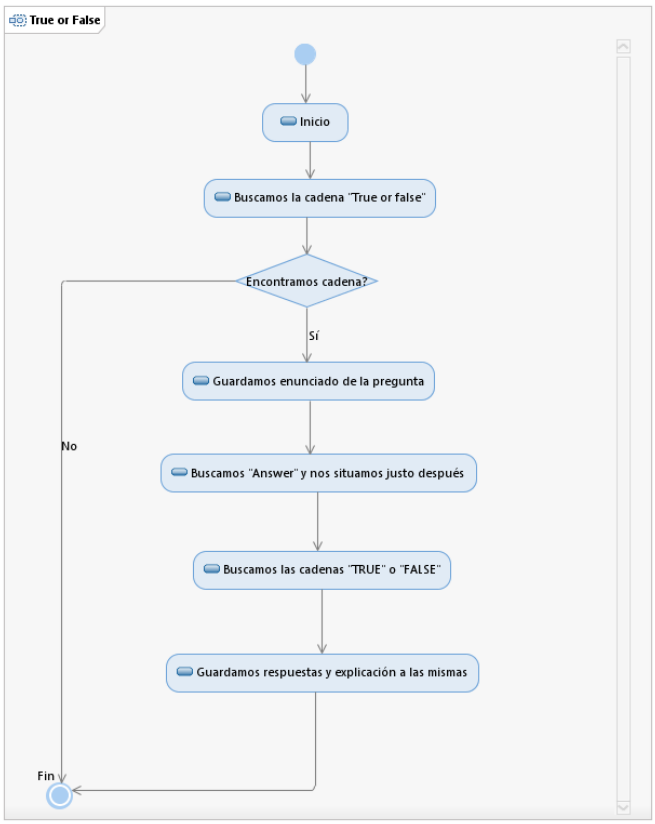


Figura 7.5: Diagrama de actividad ejercicio “True or false”

contrar la cadena “True or false”, nos situamos en el carácter posterior, donde empieza el enunciado. Tras extraer el enunciado, buscamos la cadena “Answer:”, tras la cual comienza la respuesta al ejercicio, que será un “TRUE” o un “FALSE”, seguido de la explicación pertinente. Para separar la respuesta de la explicación deberá utilizarse siempre un caracter *dos puntos* seguido de un espacio en blanco. El procedimiento se puede ver en la Figura 7.5

7.5.1.3. Fill in the gaps

Este ejercicio trata sobre rellenar los huecos de un texto con palabras que figurarán al final del mismo. A tener muy en cuenta en este ejercicio es que los caracteres salto de línea no se deben usar entre medias del propio texto, dado que el parseador buscará un caracter salto de línea para identificar el fin del mismo. En el texto debemos identificar el número de huecos que tenemos (identificados con varios caracteres “_”), para identificar posteriormente el número de palabras a utilizar y su orden. Tras ello, en la línea siguiente de la que contiene la cadena “Answer:”, comenzarán las palabras. Las x primeras,

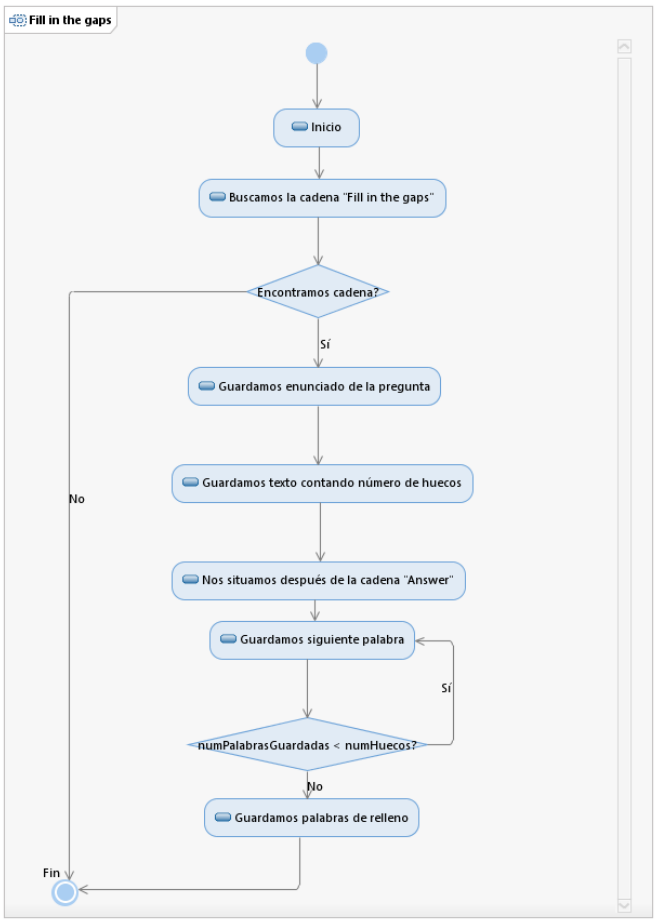


Figura 7.6: Diagrama de actividad ejercicio "Fill in the gaps"

siendo x el número de huecos que tiene el texto quedarán identificadas como las correctas con el orden adecuado y las posteriores serán las de relleno. Lo podemos ver en la Figura 7.6

7.5.1.4. Pair Words

Pair Words es un juego que consiste en unir términos relacionados entre sí dos a dos, como se muestra en la figura 7.7.

Para extraer los datos de este tipo de ejercicios, debemos primero saber cuantos pares de términos hay que casar en el ejercicio. Para ello, tenemos que buscar el literal "Answer:" y contar los saltos de línea hasta encontrarlo -habiendo extraído previamente el enunciado de la misma manera que en los ejercicios anteriores-. En una misma línea, las dos expresiones van separadas por un caracter tabulador "\t" y, a la hora de guardarlas, se irían

4.- Pair words

Match the characters with their roles in the story

1. Dr Jekyll The narrator

2. Mr Hyde The beast inside of men

3. Mr Utterson The scientist

Answer: 1-C, 2-B, 3-A

Figura 7.7: Ejemplo de ejercicio "Pair Words"

guardando separadas en dos arrays, para diferenciar los términos que vienen a la izquierda con los que vienen a la derecha. Una vez hecho esto, tras la cadena "Answer:", hay que separar los pares por comas e identificar en cada par de elementos los contenidos en ambos arrays de palabras, de cara a las inserciones concretas en la base de datos. Lo podemos ver en la Figura 7.8.

7.5.1.5. Direct questions

Este tipo de ejercicios son preguntas directas, con sus consiguientes respuestas.

El parseo des este tipo de ejercicios consiste en extraer la pregunta, sabiendo que esta siempre debe terminar en signo de interrogación y su correspondiente respuesta. Justo después de la pregunta, precediendo la respuesta, vendrá la cadena "Answer:". Justo después vendrá la respuesta a cada pregunta.

7.5.1.6. Order sentences

En este juego tenemos una serie de frases que hay que ordenar, siguiendo el criterio que nos indique el enunciado.

Para extraer los datos de este ejercicio, necesitamos dos arrays; uno en el que guardar las frases tal cual vienen al principio y el segundo donde se guardarán estos mismos elementos ya ordenados. En el documento vienen según se puede ver en la figura 7.9.

El funcionamiento es similar al de anteriores juegos. Localizar el número de saltos de línea para averiguar el número de frases a ordenar. Finalmente, a partir de la línea siguiente que aquella donde se encuentra la cadena "Answer:", encontraremos las frases ordenadas sabiendo ya el número de ellas gracias a lo calculado previamente. Lo podemos ver en la Figura 7.10.

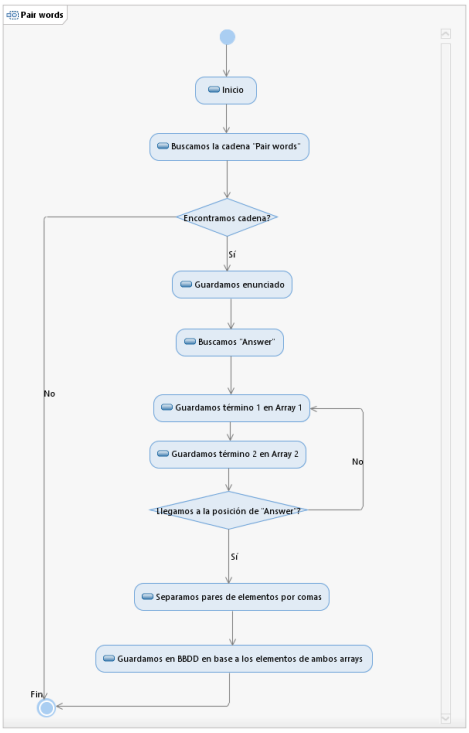


Figura 7.8: Diagrama de actividad ejercicio "Pair Words"

7.5.1.7. Recognise the text

El ejercicio *Recognise the text* consiste en enlazar un fragmento de texto con un tema y con un título referente a ellos.

Con este ejercicio tuvimos muchos problemas por la dificultad de enlazar los términos en grupos de 3, y de naturalezas tan diferentes -Fragmento de texto, tema y título- y no conseguimos llegar a una versión lo suficientemente estable como para poderla dar por terminada. Este aspecto es algo a mejorar de cara a futuro.

7.6. Consideraciones finales

Como hemos podido ver, el parseador de documentos Word que hemos desarrollado tiene como base el reconocimiento de patrones de texto y de determinados caracteres separadores de frases o términos. Esto supone que el formato y la estructura de los documentos tiene que ser muy estricta. El documento a parsear no puede incluir saltos de línea extra o tabulaciones para separar párrafos o cadenas de texto. Los usuarios deben únicamente coger el documento modelo y rellenar el espacio dedicado a los ejercicios.

7.- Order sentences

Order the next sentences:

1. A document saying that if something happens to Dr Jekyll, this Doctor's estate will be of Mr Hyde's property.

2. Dr Jekyll's narrative is read by Mr Utterson.

3. Dr Jekyll writes to Mr Utterson saying that his friendship with Dr Lanyon is now over.

4. A letter to Mr Utterson is retrieved from the old man corpse.

5. Mr Utterson reads a document written by Dr Jekyll with instructions on the reading order of certain documents.

6. Mr Utterson opens Lanyon's envelope and finds another envelope.

Answer:

1. A document saying that if something happens to Dr Jekyll, this Doctor's estate will be of Mr Hyde's property.

2. A letter to Mr Utterson is retrieved from the old man corpse.

3. Dr Jekyll writes to Mr Utterson saying that his friendship with Dr Lanyon is now ver.

4. Mr Utterson opens Lanyon's envelope and finds another envelope.

5. Mr Utterson reads a document written by Dr Jekyll with instructions on the reading order of certain documents.

Figura 7.9: Ejemplo de ejercicio "Order sentences"

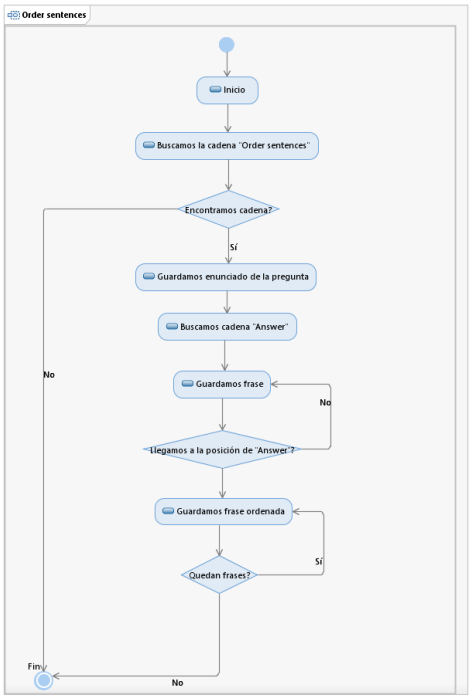


Figura 7.10: Diagrama de actividad ejercicio "Order sentences"

Capítulo 8

Estructura de la base de datos

UNIX es simple. Sólo necesita un genio para entender su simplicidad

Dennis Ritchie, científico computacional que colaboró en el desarrollo de UNIX

En cuanto a la base de datos, esta contiene una gran cantidad de tablas, las cuales tienen una red de relaciones compleja. Su estructura no es muy intuitiva si no se conoce bien la aplicación, y entenderla y adaptarnos a ella fue una de las partes más difíciles del desarrollo, por lo que nos gustaría dejar una breve explicación de su estructura para que en el futuro sea más sencillo continuar con el proyecto de *Quizz*. Podemos encontrar el modelo de la base de datos que vamos a proceder a describir en la figura **??**

Empezaremos por describir las tablas que ya formaban parte de la base de datos de la aplicación Android de *Quizz*, de las cuales nosotros partimos inicialmente:

- La tabla *class*, que contiene los datos de una clase y el profesor de esta.
- La tabla *user*, que contiene los datos de un usuario, incluyendo si este es estudiante o profesor.
- La tabla *student_ classroom*, que se encarga de relacionar a los estudiantes con sus respectivas clases, establece una relación de *n a n* entre las tablas *user* y *class*.
- La tabla *section*, que contiene la información de una sección y la clase a la que pertenece. La relación entre las tablas *class* y *section* es de *1 a n*.
- La tabla *content_ section*, que se encarga de relacionar las secciones con los contenidos de estas. La relación entre las tablas *section* y *content* es de *1 a n*.

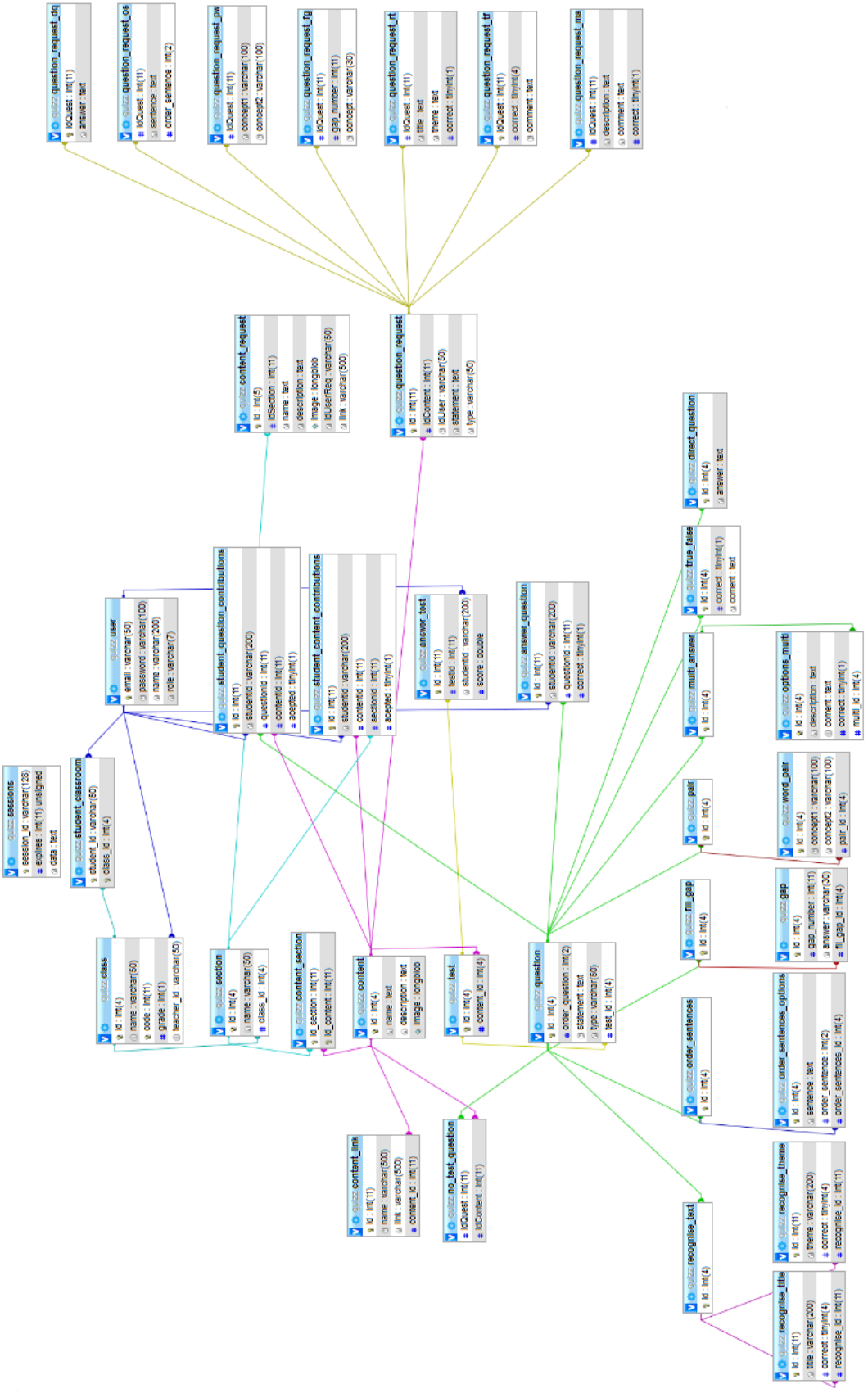


Figura 8.1: Modelo final de la base de datos

- La tabla *content*, que contiene la información de un contenido, en relación *1 a 1* con esta tabla también tenemos la tabla *content_ link*, la cual contiene el link de dicho contenido.

- La tabla *test*, la cual es usada para relacionar un grupo de ejercicios en concreto y asignarlos a un contenido. La relación de las tablas *content* y *test* es de *1 a n*, y las de las tablas *test* y *question* es de *1 a n*.

- La tabla *question*, que contiene los datos de la parte común de un ejercicio (enunciado y tipo de ejercicio) y el test y la posición en este en el que se encuentra la pregunta. Dependiendo del tipo de ejercicio que sea, a partir de aquí, se relacionará esta tabla con las correspondientes a la parte específica de ese tipo de ejercicio.

- Las tablas *recognise_ text*, *order_ sentences*, *fill_ gap*, *pair* y *multi_ anwser* se encargan de relacionar al ejercicio (tabla question) con la parte específica de su tipo de ejercicio, que son, en el mismo orden: *Recognise the text*, *fill in the gap*, *pair words*, y *multi anwser*. Estos tipos de ejercicios tienen en común que pueden tener varias respuestas, entre las cuales una y varias son correctas, por lo que se usa esta tabla para establecer la relación *1 a n* entre la tabla *question* (parte general del ejercicio) y las tablas correspondientes a las partes específicas.

- Las tablas que contienen la información específica de estos tipos de ejercicios con varias respuestas posibles son: *recognise_ title* y *recognise_ theme* para el tipo de ejercicio *recognise the text*, *order_ sentences_ option* para el tipo de ejercicio *order sentences*, *gap* para el tipo de ejercicio *fill in the gap*, *word_ pair* para el tipo de ejercicio *pair words* y finalmente *options_ multi* para el tipo de ejercicio *multiple anwser*.

- Las tablas *true_ false* y *direct_ question* contienen la información de la parte específica de los ejercicios de tipo *true or false* y *direct question* respectivamente. Estos tipos de ejercicio, al estar formados de una única respuesta, tienen una relación *1 a 1* con la tabla *question*.

Estas son todas las tablas que componían la base de datos de *Quizz* antes del inicio de nuestro proyecto, a medida que avanzabamos en este, nos dimos cuenta de que necesitábamos más tablas para conseguir alcanzar los objetivos propuestos así que añadimos las siguientes tablas:

- La tabla *sessions* se encarga de almacenar los datos de la sesiones de los usuarios de la aplicación web.

- La tabla *anwser_ test* se encarga de almacenar los datos de la realización de un test desde la aplicación Android (la nota final de éste) y que usuario lo hace, para posteriormente generar estadísticas.

- La tabla *anwser_ question* se encarga de almacenar los datos de la realización de un ejercicio desde la aplicación Android (si se ha realizado correctamente o no) y que usuario lo realiza, para posteriormente generar estadísticas.

- Las tablas *student_ question_ contributions* y *student_ content_ contributions* se encargan de almacenar los datos de las contribuciones de ejercicios y contenidos respectivamente por parte de un alumno desde la aplicación web. Estos datos se utilizan para generar estadísticas y dar información al profesor sobre sus clases.

- La tabla *content_ request* se encarga de almacenar los datos de las solicitudes de contenido realizadas desde la aplicación web.

- La tabla *question_ request* se encarga de almacenar los datos de la parte general de una solicitud de ejercicio realizada desde la aplicación web. Para almacenar la parte específica del ejercicio, según su tipo tenemos las tablas: *question_ request_ tf* para el tipo de ejercicio *true or false*, *question_ request_ dq* para el tipo de ejercicio *direct question*, *question_ request_ os* para el tipo de ejercicio *order sentences*, *question_ request_ fg* para el tipo de ejercicio *fill in the gap*, *question_ request_ rt* para el tipo de ejercicio *recognise the text*, *question_ request_ ma* para el tipo de ejercicio *multi anwser* y *question_ request_ pw* para el tipo de ejercicio *pair words*. Estas tablas tienen una relación de *n a 1* con la tabla *question request*.

- Finalmente tenemos la tabla *no_ test_ question* que se encarga de almacenar los ejercicios que se encuentran sin test asignado, es decir, cuya solicitud de ejercicio enviada por el alumno ya ha sido aprobada por el profesor, pero no ha sido añadido a ningun test.

Capítulo 9

Conclusiones y trabajo futuro

El software es un gas: se expande hasta llenar su contenedor

Nathan Myhrvold, ex-gerente de tecnología de Microsoft

9.1. Conclusiones

Podemos concluir, una vez finalizado el proyecto, que los dos objetivos principales que nos propusimos al comienzo del proyecto han sido cumplidos:

- El primero era desarrollar una aplicación web completamente responsive, complementaria a la aplicación móvil de *Quizz*, que permitiese la subida de nuevos contenidos y la visualización de los datos de sus clases y alumnos al profesor.
- El segundo era desarrollar un parseador que permitiese al profesor añadir contenidos a la aplicación directamente desde documentos word con una estructura predefinida.

Estos dos puntos principales han sido cumplidos, además de esto, gracias a la aplicación web desarrollada y al parseador, un profesor puede crear sus propias clases y contenidos de cualquier asignatura, por lo que *Quizz* se podrá expandir a cualquier tipo de materia, y ya no dependerá de los desarrolladores para crear dichas clases y contenidos, sino que será completamente autónoma.

También queremos destacar que este proyecto nos ha permitido investigar y aprender a usar un amplio abanico de tecnologías web, siendo todas estas de software libre. También hemos podido ganar experiencia en la gestión y el desarrollo de proyectos, y en cómo gestionar el trabajo en equipo.

Queremos añadir que para el desarrollo de nuestra aplicación web y nuestro parser, también hemos tenido que trabajar sobre una estructura de datos

ya diseñada por otros alumnos, por lo que la hemos tenido que estudiar y modificar de manera que no afectase al trabajo existente, lo cual ha añadido complejidad a nuestro proyecto, ya que no éramos libres de trabajar con esta estructura a nuestro antojo. Finalmente conseguimos no solo trabajar sobre esta estructura de datos, sino mejorarla para ampliar sus posibilidades de uso.

Nuestro trabajo está disponible en el siguiente repositorio público `https://github.com/notnavas/QuizzWeb`, así como en el siguiente privado `https://github.com/NILGroup/TFG-1718-Quizz`. La aplicación se está ejecutando en el siguiente servidor `http://tot.fdi.ucm.es:3000`.

Por último, el desarrollo de este proyecto también nos ha servido para aprender en otros aspectos del desarrollo, como son el diseño de una aplicación web para hacerla fácil e intuitiva y completamente responsive, la seguridad de la aplicación para controlar que cada usuario solo tuviese acceso a los datos que le corresponden, o el desarrollo de toda la aplicación para hacerla plenamente funcional y consistente.

9.2. Trabajo futuro

Dado que la duración del proyecto ha sido de menos de un año, no hemos conseguido implementar algunas de las características que nos parecen interesantes para un mejor funcionamiento de nuestra aplicación y que por lo tanto dejaremos marcadas como trabajo futuro.

En cuanto a la aplicación web hay una serie de funcionalidades que nos parece que podrían mejorar significativamente el funcionamiento y usabilidad de ésta:

- Una de estas características que nos parece muy interesante es el tener la posibilidad de copiar contenido de una clase a otra, para que si el profesor lo desea pueda intercambiar contenido entre sus clases, empezamos con la implementación de esta funcionalidad y dejamos implementado tanto sus vistas como su formulario para conseguir los datos necesarios para esta funcionalidad, pero no conseguimos acabarla a tiempo, por lo que la dejamos como trabajo para futuras versiones.

- Otra características interesante sería que el profesor tuviera la capacidad de modificar las solicitudes recibidas por los alumnos antes de aceptarlas, para poder corregir pequeños detalles o aprovechar este contenido.

- También sería buena idea aumentar la capacidad del sistema de subida de contenido, para que no sólo fuese posible subir contenido, sino también modificarlo y eliminarlo.

- Como explicamos anteriormente, hubo un tipo de ejercicio que no fuimos capaces de parsear correctamente. Es prioritario de cara al futuro terminar este trabajo.

- Otra idea en la que pensamos, la cual llevaría una cantidad de trabajo bastante amplia, sería crear un sistema contrario al parseador, que en vez de pasar contenido de un word a la base de datos, pudiese descargar el contenido de la base de datos a un word, como forma de conseguir apuntes y ejercicios realizados por los alumnos. Esto sería una característica que daría a la aplicación algo único, que no se encuentra en otras aplicaciones de este tipo.

En cuanto la aplicación Android, no tuvimos tiempo de modificarla ya que la aplicación web y el parseador fueron nuestras prioridades, por lo que quedan pendientes algunos puntos que ayudarían al mejor funcionamiento de la aplicación:

- Uno de estos puntos, el cual sería prioritario, sería el modificar el sistema de imágenes de la aplicación Android, ya que actualmente, se guardan como un tipo de archivo *BLOB* en la base de datos, lo cual la hace pesada y hace que tratar este tipo de imágenes desde la aplicación web y el parsearla sea algo muy complicado. En vez de usar este tipo de imagenes, solo seria necesario guardar el nombre de la imagen, el cual desde la aplicación web se genera automáticamente al subirla y mostrar la imagen alojada en la aplicación web desde su ruta. La subida, alojamiento y renombrado de imágenes ya han sido implementado en la aplicación web, pero no está activado para no interferir con el correcto funcionamiento de la aplicación Android.

- Otro punto es el modificar la aplicación Android para que al realizar un test o un ejercicio se rellenen las tablas de la base de datos *answer_ test* y *anwser_ question* ya que en este punto y dado que no hemos modificado la aplicación Android, los datos de estas tablas han sido añadidos manualmente para realizar las gráficas de estadísticas.

- Por otro lado, resultaría útil la inclusión de nuevos tipos de ejercicios para ampliar la funcionalidad de la aplicación y dar más posibilidades a los profesores a la hora de crear los test de los contenidos. También sería importante extrapolar el uso de esta aplicación móvil para que sea accesible en otras plataformas, como pueden ser *iOS* o *Windows Phone*.

Capítulo 10

Conclusions and future work

Hardware: las partes de un ordenador
que pueden ser pateadas

Jeff Pesis

10.1. Conclusions

We can conclude, once the project is finished, that the two first goals which we intended to achieve when we started the project are done:

- The first one, was to develop a fully responsive web application, which could complement the Android application *Quizz*, and allowed the upload of new contents and the visualization of the data from their classes and students to the teachers.
- The second one was to develop a parser which allowed the teacher to add new contents to the application directly taken from word documents with a predefined structure.

These two main points have been met, in addition to this, thanks to the developed web application and the parser, a teacher can create their own classes and contents of any subject, so that *Quizz* can be expanded to any type of subject, and it will not depend on the developers to create such classes and contents but it will be completely autonomous.

We also want to emphasize that this project has allowed us to research and learn to use a wide range of web technologies, all of which are free software. We have also been able to gain experience in the management and development of projects, and gave us experience about how to manage the teamwork.

We want to add that for the development of our web application and our parser, we have also had to work with a data structure already designed by

other students, so we have had to study and modify it in a way that does not affect the existing work. This fact has added complexity to our project, since we were not free to work with this structure at our whim. Finally we managed not only to work with this data structure, but to improve it to expand its possibilities of use.

Our job is available in this public repository `https://github.com/notnavas/QuizzWeb` and also in the next private one `https://github.com/NILGroup/TFG-1718-Quizz`. The application is running on the next server `http://tot.fdi.ucm.es:3000`.

Finally, the development of this project has also helped us to learn in other aspects of development, such as the design of a web application to make it easy and intuitive and completely responsive, the security of the application to control that each user only had access to the data that he is allowed to access, or the development of the entire application to make it fully functional and consistent.

10.2. Future work

Given that the duration of the project has been less than a year, we have not managed to implement some of the features that we find interesting for a better functioning of our application and that will therefore be marked as future work.

Regarding the web application there are a number of functionalities that we believe could significantly improve the functionality and usability of our application:

- One of these features that we find very interesting is having the posibility to copy content from one class to another, so that if the teacher wishes, he can exchange content between his classes. We started with the implementation of this functionality and we implemented both its views and its forms to get the necessary data for this functionality, but we did not manage to finish it on time, so we left it as work for future versions.

- Another interesting feature would be that the teacher had the ability to modify the requests received by the students before accepting them, in order to correct small details or use parts of this content.

- It would also be a good idea to increase the capacity of the content upload system, so that not only could content be uploaded, but also modified and eliminated.

- Another idea in which we thought, which would take a large amount of work, would be to create a system opposed to the parser, which instead

of passing content from a word to the database, could download the contents from the database to a word document, as a way to get notes and exercises made by students. This would be a feature that would give the application something unique, which is not found in other applications of this type.

Related the Android application, we did not have time to modify it since the web application and the parser were our priorities, so there are some points pending that would help the best functioning of the application:

- One of these points, which would be a priority, would be to modify the image system of the Android application, since currently, they are saved as a *BLOB* file in the database, which makes the database heavy and makes hard to work with this images. Instead of using this type of images, it would only be necessary to save the name of the image, which is generated automatically from the web application when uploading it and display the image hosted in the web application from its route. The upload, hosting and renaming of images have already been implemented in the web application, but it is not activated in order not to interfere with the correct functioning of the Android application.
- Another point is the modification of the Android application so that when performing a test or an exercise, the tables of the *answer_ test* and *anwser_ question* database are filled in, because at this point and given that the Android application has not been modified, the data of these tables have been added manually to make the statistics graphs.
- On the other hand, it would be useful to include new types of exercises to extend the functionality of the application and give more possibilities to teachers when creating content tests.
- It would also be important to extrapolate the use of this mobile application to be accessible on other platforms, such as *iOS* or *Windows Phone*.

Trabajo individual

El talento gana partidos, pero el trabajo en equipo y la inteligencia ganan campeonatos

Michael Jordan, ex-jugador de baloncesto.

Álvaro Navas Sánchez

En primer lugar, realicé un estudio exhaustivo de cómo estaba programada la aplicación Android de la que partimos, para comprender en profundidad la base de datos, la cual era bastante complicada, y poder modificarla y trabajar con ella sin afectar a la aplicación Android existente.

Una vez conseguido esto, empecé con el desarrollo de la aplicación web en node, utilizando *Javascript* y el framework *Express.js*. Posteriormente conecté la aplicación web con la base de datos existente de *Quizz* mediante *Node* y *SQL* para poder realizar un sistema de login y sesiones compatibles con dicha base de datos. Inicialmente la web estaba solo pensada para uso del profesor, pero al empezar el desarrollo, vimos que sería útil que los alumnos tuviesen acceso a la web para poder realizar subidas de contenido, las cuales el profesor podría aceptar o rechazar.

De tal forma se dividió la funcionalidad de la web en función del tipo de usuario que se loguease, profesor o alumno.

Una vez establecida la conectividad y el sistema de sesiones, empecé con la interfaz gráfica de la aplicación. Para esto decidí usar una librería de código libre, basada en *Bootstrap*, llamada *AdminLTE*. *AdminLTE* contiene un sistema de vistas y componentes predefinidos y compatibles con dispositivos móviles, los cuales no eran del todo compatibles con *Express.js*, el framework que había decidido usar, ni con su sistema de plantillas *EJS*, por lo cual decidí integrar manualmente dicha librería y la modifiqué para que fuese compatible.

Una vez definida la interfaz, siendo esta completamente compatible con dispositivos móviles, comencé con el desarrollo de las funcionalidades de la web y las vistas de estas:

- Para el profesor:
  - Visualizar todas sus clases, contenidos y alumnos.
  - Poder acceder a los alumnos y clases para visualizar sus perfiles y estadísticas.
  - Añadir contenido a clases existentes de forma manual.
  - Crear nuevas clases.
  - Visualizar y, aceptar o rechazar solicitudes de contenidos o ejercicios enviados por alumnos.
  - Generar test aleatorios con los ejercicios previamente aceptados.
- Para el alumno:
  - Facilitar formularios para la creación de solicitudes de contenido y de ejercicios, según su tipo.
  - Visualizar sus estadísticas para ver su progreso en la asignatura.

Para crear un canal de comunicación interno, con fines tanto educativos como lúdicos, y llamar al uso de la web, también implementé un chat interno, para cada clase, al cual tienen acceso tanto profesores como alumnos, para esto usé la librería de codigo libre *socket.io*.

En cuanto a la base de datos, para implementar las nuevas funcionalidades añadí nuevas tablas, siempre teniendo en cuenta no modificar la estructura general para no afectar al funcionamiento de la aplicación web.

En este punto teníamos una versión funcional de la web, compatible con la aplicación de Android y con una forma consistente de subir contenidos, por lo que los objetivos iniciales del proyecto en relación con la aplicación web habían sido cumplidos.

En cuanto a todas las tecnologías usadas, la mayoría no las conocía, así que realice una búsqueda de todas ellas y tras decidir que eran las que mejor encajaban con el proyecto aprendí a usarlas para posteriormente integrarlas a la aplicación web.

Durante todo el desarrollo del proyecto he redactado las siguientes partes:

- La parte correspondiente al punto de partida de la aplicación.
- La parte en la que se explica el desarrollo y la implementación del portal web.
- Parte del punto que explica la arquitectura de la aplicación.
- La parte referente a las conclusiones y el trabajo futuro, tanto en castellano como en inglés.
- Parte del capítulo referente a las tecnologías utilizadas.
- Parte de la bibliografía de este trabajo.

Iván Quirós Fernández-Montes

Al comienzo de este proyecto me dediqué, al igual que mi compañero, a realizar un estudio exhaustivo de la aplicación *Quizz*.

Para ello, tuvimos que descargarnos la aplicación de la *Google Play Store*, probarla para comprender su funcionamiento, revisar a fondo la memoria de los compañeros que la desarrollaron el curso pasado y entender correctamente el modelo de datos de la misma.

Tras esto, junto con nuestro director de TFG, realizamos una evaluación de las necesidades que tenían los usarios de la misma de cara a poder abordar los desarrollos necesarios.

Una vez quedaron definidas las necesidades, llegó el momento de dividirse el trabajo. Llegamos a la conclusión de que debíamos dividir el trabajo en dos aspectos fundamentales: el portal web y el parseador de documentos.

Dentro de estas dos necesidades que identificamos, yo me encargué del desarrollo del parseador de documentos Word.

Como primera tarea, me tocó realizar un estudio de las posibilidades que tenía para llevar a cabo esta tarea. Desde lenguaje de programación (como se ha explicado a lo largo de esta memoria, por diferentes motivos, se decidió hacerlo en *JavaScript*), pasando por una revisión a fondo de un importante número de ejemplos de documentos Word. Todo esto para tratar de encontrar la mejor manera de implementar este parseador. Una vez hecho este estudio, me dediqué a buscar por distintas comunidades de desarrolladores acerca de las mejores herramientas (tanto entornos de desarrollo, como bibliotecas o *Frameworks*) para llevar a cabo este desarrollo. Entre otros sitios, se buscó información en los siguientes:

- *Stackoverflow* (stackoverflow, 2018) es el sitio web para desarrolladores informáticos más conocido y utilizado del mundo. En él, los desarrolladores pueden consultar sus dudas o resolver dudas de otros desarrolladores. Tiene versiones propias para varios idiomas, con el castellano entre ellos.

- *GitHub* es un sitio web que permite alojar proyectos de cualquier tipo (código en cualquier lenguaje, en el caso de una aplicación, documentos...) que utiliza el sistema de control de versiones *Git*.

- *npmjs* es el sitio web oficial del manejador de paquetes *npm*, que es el utilizado por defecto para *Node.js*. En este sitio busqué información sobre la mejor solución para implementar el parseador.

Una vez terminada la parte del parseador, realicé, en colaboración con mi compañero, la integración del mismo con el portal web; teniendo en cuenta que, como se ha explicado a lo largo de esta memoria, sería el profesor quien

realizaría las inserciones de los documentos y por tanto había que integrarlo con su parte.

Por otra parte, me encargué de los siguientes puntos en la presente memoria:

- El capítulo de introducción de la memoria.
- La parte referente al trabajo relacionado.
- El capítulo referente al parseador de documentos.
- Parte del punto que habla de la explicación de la arquitectura de aplicación.
- Parte del trabajo que trata sobre las tecnologías usadas.
- Parte de la bibliografía de este trabajo

Aparte de esto, fui el encargado de maquetar el presente documento utilizando LaTeX.

Bibliografía

Y así, del mucho leer y del poco dormir,
se le secó el celebro de manera que vino
a perder el juicio.

Miguel de Cervantes Saavedra

Apache. Apache poi - hwpf and xwpf - java api to handle microsoft word files. 2010. Disponible en `http://poi.apache.org/document/index.html` (último acceso, Mayo, 2018).

Apache. The apache software foundation. 2018. Disponible en `https://www.apache.org` (último acceso, Mayo, 2018).

Canny, S. python-docx. 2013. Disponible en `https://python-docx.readthedocs.io/en/latest` (último acceso, Mayo, 2018).

web docs, M. Trabajando con json. 2017. Disponible en `https://developer.mozilla.org/es/docs/Learn/JavaScript/Objects/JSON` (último acceso, Mayo, 2018).

lalalic GitHub. docx4js. 2017. Disponible en `https://github.com/lalalic/docx4js` (último acceso, Mayo, 2018).

GNU. Licencia pública general de gnu. 2018. Disponible en `https://www.gnu.org/licenses/licenses.es.html#GPL` (último acceso, Mayo, 2018).

Linux, M. Xml ¿qué es? 2017. Disponible en `http://www.mundolinux.info/que-es-xml.htm` (último acceso, Mayo, 2018).

ManuelMontenegro. Manuelmontenegro, desarrollo web. 2018. Disponible en `https://manuelmontenegro.github.io/AW-2017-18/` (último acceso, Mayo, 2018).

Milán, V. Crear tabs o pestañas con jquery y css. 2010. Disponible en `https://www.lawebera.es/como-hacer/ejemplos-jquery/tabs-pestanas-con-jquery-css.php` (último acceso, Mayo, 2018).

PORTA, M. T. Una plataforma móvil para la enseñanza de literatura inglesa. 2016. Disponible en http://eprints.ucm.es/38677/ (último acceso, Mayo, 2018).

ÁNGELA ROCÍO CAMARGO SÁNCHEZ y FERNÁNDEZ, A. J. M. Quizz: Una plataforma móvil para la enseñanza de literatura inglesa. 2017. Disponible en http://eprints.ucm.es/44408/ (último acceso, Mayo, 2018).

STACKOVERFLOW. stackoverflow, programacion. 2018. Disponible en https://es.stackoverflow.com/ (último acceso, Mayo, 2018).

WIKIPEDIA (CSS). Entrada: "CSS". Disponible en https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada (último acceso, Mayo, 2018).

WIKIPEDIA (DOM). Entrada: "Document Object Model". Disponible en https://es.wikipedia.org/wiki/Document_Object_Model (último acceso, Mayo, 2018).

WIKIPEDIA (IDE). Entrada: "Entorno de desarrollo integrado". Disponible en https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado (último acceso, Mayo, 2018).

WIKIPEDIA (HTML5). Entrada: "HTML5". Disponible en https://es.wikipedia.org/wiki/HTML5 (último acceso, Mayo, 2018).

WIKIPEDIA (ClientSide). Entrada: "Lado del cliente". Disponible en https://es.wikipedia.org/wiki/Lado_del_cliente (último acceso, Mayo, 2018).

WIKIPEDIA (Material Design). Entrada: "MaterialDesign". Disponible en https://es.wikipedia.org/wiki/Material_design (último acceso, Mayo, 2018).

WIKIPEDIA (InfoPath). Entrada: "Microsoft InfoPath". Disponible en https://es.wikipedia.org/wiki/Microsoft_InfoPath (último acceso, Mayo, 2018).

WIKIPEDIA (TeX). Entrada: "TeX". Disponible en https://es.wikipedia.org/wiki/TeX (último acceso, Mayo, 2018).

WIKIPEDIA (Xampp). Entrada: "Xampp". Disponible en https://es.wikipedia.org/wiki/XAMPP (último acceso, Mayo, 2018).

WIKIPEDIA (XHTML). Entrada: "XHTML". Disponible en https://es.wikipedia.org/wiki/XHTML (último acceso, Mayo, 2018).

XATAKA. Chrome os, análisis. 2014. Disponible en https://www.xataka.com/analisis/chrome-os-analisis (último acceso, Mayo, 2018).

Xataka. Atentos, fans de 'juego de tronos': Duolingo os enseña a hablar en alto valirio. 2017. Disponible en `https://www.xataka.com/cine-y-tv/atentos-fans-de-juego-de-tronos-duolingo-os-ensena-a-hablar-en-alto-valirio` (último acceso, Mayo, 2018).

–¿Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.

Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes

–Buena está – dijo Sancho –; fírmela vuestra merced.
–No es menester firmarla – dijo Don Quijote–,
sino solamente poner mi rúbrica.

Primera parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes